THESIS

IMPLEMENTATION OF A MULTI-ROBOT COVERAGE ALGORITHM ON A TWO-DIMENSIONAL, GRID-BASED ENVIRONMENT

by

Jo-Wen Huang

June 2017

Thesis Advisor:	Brian Bingham
Second Reader:	Wei Kang

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2017	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE IMPLEMENTATION OF A MULTI-ROBOT COVERAGE ALGORITHM ON A TWO-DIMENSIONAL, GRID-BASED ENVIRONMENT	**5. FUNDING NUMBERS**
6. AUTHOR(S) Jo-Wen Huang	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

With the development and advancement in the technology of control and multi-robot systems, robot agents are likely to take over mine countermeasure (MCM) missions one day. The path planning coverage algorithm is an essential topic for research; the combination of an efficient algorithm and accurate sensors can save time and human lives. The objective of this work is to implement a path planning coverage algorithm for a multi-robot system in a two-dimensional, grid-based environment. We assess the applicability of a topology-based algorithm to the MCM mission. First, we provide an overview of multi-robot coverage algorithms. Second, we select one algorithm, analyze it, and test its performance. Then the algorithm is evaluated in nine experiments using different numbers of robots and obstacles. Finally, the results are assessed by how much time the steps took and how many free points are not visited when the algorithm is finished. The outcome indicates that efficiency decreases as the number of robots or obstacles increases. This thesis concludes with recommendations for ways to improve the efficiency of the algorithm as well as how to perform the experiments cost effectively in a real environment.

14. SUBJECT TERMS path planning, exact cellular decompositions, multi-robot coverage algorithm, mine countermeasure	**15. NUMBER OF PAGES** 105
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

THIS PAGE INTENTIONALLY LEFT BLANK

IMPLEMENTATION OF A MULTI-ROBOT COVERAGE ALGORITHM ON A TWO-DIMENSIONAL, GRID-BASED ENVIRONMENT

Jo-Wen Huang
Lieutenant, Taiwan Navy
B.S., Chinese Naval Academy, 2008

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2017

Approved by: Brian Bingham
Thesis Advisor

Wei Kang
Second Reader

Garth V. Hobson
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

With the development and advancement in the technology of control and multi-robot systems, robot agents are likely to take over mine countermeasure (MCM) missions one day. The path planning coverage algorithm is an essential topic for research; the combination of an efficient algorithm and accurate sensors can save time and human lives. The objective of this work is to implement a path planning coverage algorithm for a multi-robot system in a two-dimensional, grid-based environment. We assess the applicability of a topology-based algorithm to the MCM mission. First, we provide an overview of multi-robot coverage algorithms. Second, we select one algorithm, analyze it, and test its performance. Then the algorithm is evaluated in nine experiments using different numbers of robots and obstacles. Finally, the results are assessed by how much time the steps took and how many free points are not visited when the algorithm is finished. The outcome indicates that efficiency decreases as the number of robots or obstacles increases. This thesis concludes with recommendations for ways to improve the efficiency of the algorithm as well as how to perform the experiments cost effectively in a real environment.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION..1
- **A. PURPOSE OF RESEARCH 1**
- **B. APPROACH 2**
  - **1. Search for Related Coverage Algorithm.....................................2**
  - **2. Coverage Algorithm of Interest 6**
  - **3. Develop MATLAB Code from Pseudo Code 6**
  - **4. Simulation Experiment 7**
- **C. CONTRIBUTIONS..7**
- **D. ORGANIZATION OF STUDY 8**

II. LITERATURE REVIEW 9
- **A. SPECTRUM OF ROBOT CONTROL ARCHITECTURES FOR ALGORITHMS 10**
  - **1. Behavior–Based Approach 10**
  - **2. System Theory–Based Approach 10**
- **B. STATE OF THE ART 13**
- **C. ALGORITHMS SUMMARY 21**
- **D. DETERMINE ALGORITHMS FOR THE MCM TASK...................24**

III. ALGORITHM DEVELOPMENT 25
- **A. GENERAL TOPOLOGY–BASED ALGORITHM 25**
- **B. NORMAL MODE 26**
  - **1. Description of the Basic Concept of Normal Mode..................27**
  - **2. Grid–Based Implementation...32**
- **C. WRAP MODE..35**
  - **1. Description of the Basic Concept of Wrap Mode......................35**
- **D. DIRECTION OF MOVEMENT IN WRAP MODE ANALYSIS (COUNTERCLOCKWISE EXAMPLE)...38**
  - **1. Basic Concept of Moving Counterclockwise 38**
  - **2. Definition of “Required Points” 39**
  - **3. Concept to Go Clockwise...53**
- **E. TWO-DIMENSIONAL GRID–BASED IMPLEMENTATION.........54**
  - **1. Normal Mode...54**
  - **2. Normal Mode + Wrap Mode...55**
- **F. DEVELOPED MATLAB CODE FOR THE ALGORITHM.............55**

IV. SIMULATION EXPERIMENT 57
- **A. DESCRIPTION OF EXPERIMENTS..57**

1. Domain Area....58
2. Obstacles....58
3. Number of Robots....60
4. Experiment Scenarios....60
5. Evaluation Methods....60
B. PREDICTION....61
C. EXPERIMENTS....62
1. One Robot with One Obstacle $(1,1)$....62
2. One Robot with Two Obstacles $(1,2)$....63
3. One Robot with Three Obstacles $(1,3)$....64
4. Two Robots with One Obstacle $(2,1)$....65
5. Two Robots with Two Obstacles $(2,2)$....67
6. Two Robots with Three Obstacles $(2,3)$....68
7. Three Robots with One Obstacle $(3,1)$....69
8. Three Robots with Two Obstacles $(3,2)$....70
9. Three Robots with Three Obstacles $(3,3)$....71
D. RESULTS AND ANALYSIS....71
1. Results of Experiments....71
2. Factors that Affect Time Steps....72
3. Factor that Affects Complete Coverage....74

V. CONCLUSION....77
A. PROS AND CONS....78
1. Idea....79
2. Scalability....79
3. Completeness of Coverage....79
B. FUTURE WORK....79
1. Monte Carlo Simulation....80
2. Improve the Algorithm....80
3. Run the Algorithm with Robotic Simulator....80
4. Appliy the Algorithm on Land Robots in MCM-like Environment....80
5. Implement the Algorithm on an AUV in a Littoral Zone....80

LIST OF REFERENCES....83

INITIAL DISTRIBUTION LIST....85

LIST OF FIGURES

Figure 1. Approximation of an Obstacle by Uniform Grids. Source: [2]. 3

Figure 2. An Efficient Path Found by the Wavefront Algorithm. Adapted from [3]. 3

Figure 3. Path of Complete Coverage Executed by Conventional Wavefront Algorithm. Source: [3]. 4

Figure 4. Example of Exact Decomposition—Trapezoidal Decomposition. Source: [4]..................5

Figure 5. Example of Semi-approximate Decomposition. Adapted from [1]. 6

Figure 6. Spectrum of Robot Control Architectures 11

Figure 7. The Result of Probability of Coverage in Histogram. Source: [11]. 15

Figure 8. Experiment Result for the Three-Target Case. Source: [15]..................21

Figure 9. Pseudo Code of Normal Mode. Source: [5]..................26

Figure 10. The Basic Concept of Defining $\mathbf{E}_{c',j}$ 28

Figure 11. Illustration of $\mathrm{N}\left(E_{c',\mathrm{j}},\ 4\mathrm{d}_r\right)$ 28

Figure 12. Illustration of Boundary of Intersection of $\mathbf{N}(\mathbf{q}_{ri}, 0.5\mathbf{d}_r)$ and $\mathbf{N(E}_{c,i}\mathbf{,}0.5\mathbf{d}_r\mathbf{)}$ 29

Figure 13. Illustration of $\mathbf{B}_{N*}$ 29

Figure 14. Illustration of Intersection of $\mathbf{N}(\mathbf{q}_{ri}, \mathbf{d}_{ob})$ and $\mathbf{N}(\mathbf{O}_k, \mathbf{d}_{ob})$ 30

Figure 15. Illustration of Intersection of $\mathbf{N}(\mathbf{q}_{ri}, 2\mathbf{d}_{ob})$ and $\mathbf{N}\left(\mathbf{E}_{c',j}, 2\mathbf{d}_{ob}\right)$ 31

Figure 16. Illustration of Choosing Next Point for $\mathbf{q}_{ri}$ 31

Figure 17. Direction that a Robot Can Turn in Original Algorithm. Source: [5]........32

Figure 18. The Diameter of Robot Footprint Is $\mathbf{d}_r$ 32

Figure 19. Possible and Uncovered Spaces for Robot $^{\#}$1 to Move (Part 1) 33

Figure 20. Possible and Uncovered Spaces for Robot $^{\#}1$ to Move (Part 2) 34

Figure 21. Pseudo Code of Wrap Code. Source: [5]. 35

Figure 22. Illustration of Redefined Intersection of $\mathrm{N}(q_{ri}, 2d_{ob})$ and $N(\mathrm{E}_{\mathrm{c,\,j}}, 2d_{ob})$ 36

Figure 23. Illustration of $\mathbf{q}_{ri}$ Equals $\mathbf{q}_{st}$ under Wrap Mode and Moving toward Existing $\mathbf{q}_{B,i}$ 37

Figure 24. Illustration of $\mathbf{q}_{st}$ Becoming $\mathbf{q}_{tr}$ as $\mathbf{q}_{B,i}$ Cannot Be Located under Wrap Mode when $\mathbf{q}_{ri}$ Moves Counterclockwise 37

Figure 25. Illustration of $\mathbf{q}_{ri}$ Entering Idle Mode if $\mathbf{q}_{ri}$ Cannot Find $\mathbf{q}_{B,i}$ When Moving Clockwise from Transition Point and Reaching $\mathbf{q}_{tr}$ Again..........38

Figure 26. Concept for $\mathbf{q}_{ri}$ to Determine a Direction...39

Figure 27. Comparison of Angles 39

Figure 28. Four Cases for Directions in which $\mathbf{q}_{ri}$ Could Move................................40

Figure 29. Required Point Is One Move...40

Figure 30. Two Different Distances as Required Points Equal Two Moves 41

Figure 31. Required Points Equal Two Moves 42

Figure 32. Neighborhood Patterns of $\mathbf{q}_{ri}$ 42

Figure 33. Intersection of SS and EP Equals Zero 43

Figure 34. Required Points Equal Three Moves and Intersection with Small Square Equals Zero 43

Figure 35. Zero Point Occupies Small Square under the Required Points Equal Three Moves Condition 44

Figure 36. Intersection of SS and EP Equals One Move...45

Figure 37. Required Points Equal to Three Moves and Intersection with Small Square Is One Move...46

Figure 38. One Point Occupies the Small Square under the Required Points Equal Three Moves Condition 46

Figure 39. Intersection of SS and EP Equals Two Moves 47

Figure 40. Required Points Equal Three Moves and Intersection with Small Square Equals Two Moves 47

Figure 41. Two Points Occupy the Small Square under the Required Points Are Three Moves Condition 48

Figure 42. Intersection of SS and EP Equals Three Moves 49

Figure 43. Required Points Equal Three Moves and the Intersection with Small Square Equals Three Moves 49

Figure 44. Three Points Occupy Small Square under the Required Points Equal Three Moves Condition 50

Figure 45. Required Points Equal Four Moves 51

Figure 46. Demonstration of Finding the Critical Point 51

Figure 47. Demonstration of Locating Next Moves 51

Figure 48. Required Points Equal Four Moves 52

Figure 49. Required Point Equals Zero 53

Figure 50. Normal Mode Performed by Two Robots 54

Figure 51. Robots Switch between Normal Mode and Wrap Mode to Complete Coverage 55

Figure 52. Unexplored Points and Neighborhood Field of Domain Area 58

Figure 53. One-Obstacle Environment 59

Figure 54. Two-Obstacle Environment 59

Figure 55. Three-Obstacle Environment 59

Figure 56. Coverage Algorithm Implemented by One Robot in One-Obstacle Environment 62

Figure 57. Coverage Algorithm Implemented by One Robot in Two-Obstacle Environment 63

Figure 58. Coverage Algorithm Implemented by One Robot in a Three-Obstacle Environment 64

Figure 59. Coverage Algorithm Implemented by Two Robots in a One-Obstacle Environment 65

Figure 60. Colleague Avoidance Behavior 66

Figure 61. Ignored Unexplored Points Visited by One Robot as the Other Robot Is Far Away..66

Figure 62. Coverage Algorithm Implemented by Two Robots in a Two-Obstacle Environment..67

Figure 63. Coverage Algorithm Implemented by Two Robots in a Three-Obstacle Environment..68

Figure 64. Coverage Algorithm Implemented by Three Robots in a One-Obstacle Environment..69

Figure 65. Coverage Algorithm Implemented by Three Robots in a One-Obstacle Environment..70

Figure 66. Coverage Algorithm Implemented by Three Robots in a One-Obstacle Environment..71

Figure 67. Ignored Unexplored Spaces Due to Colleague Avoidance Behavior 74

LIST OF TABLES

Table 1. Metrics of Probability of Detection 9

Table 2. Features of Robot Control Architectures. Adapted from [7], [8]. 12

Table 3. Performance Metrics for the Three-target Case. Source: [15]. 20

Table 4. Classification Determined by Assorted Characteristics............................22

Table 5. Nomenclature. Source: [5]. 25

Table 6. Meaning of Symbols 26

Table 7. Experiment Matrix 57

Table 8. Total Number of Free Grid Cells 61

Table 9. Lower Bound of Time Steps for Each Experiment (Min # Steps)............61

Table 10. Number of Steps Taken by Each Experiment....................72

Table 11. Final Time Steps Ratio....................72

Table 12. Coverage Completeness Ratio 75

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AUV	autonomous underwater vehicle
BBR	behavior-based robots
EP	explored points
MAC	multiple aspect coverage
MCM	mine countermeasure
MTSC	multi-robot spanning tree
SS	small square

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my husband, Meng-yen, for always encouraging me to keep my faith and pursue my dream, and for supporting our family’s daily lives. Without his sweet support, this work would have been harder. I thank my two-month-old son, Brian, for coming to be our son and giving us so much happiness that we had never had. My advisor, Professor Bingham, deserves a lot of credit for sharing his experience and instructing knowledge. I love the way he lectures. He conveys an abstract idea with “make sense” pictures, which helped me understand more about the robotics field. Last, I appreciate my second reader, Professor Wei-Kang, for sharing his professional experience in the algorithm coverage field.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

According to Google Scholar, the concept of single-robot systems has been considered since as early as 1700 while the concept of multi-robot systems emerged in the 1950s. Within the field of robotics, many applications and related materials are worth exploring. And, in this thesis, the coverage algorithm applied to both single robot and multi-robot systems is our primary topic. Coverage means the number of units within a specific area visited by robots. The coverage algorithm is a mathematical method to drive robots to achieve maximum coverage of a specified area. From the coverage perspective, using a single robot to complete coverage algorithms is no longer a challenge when the terrain in a defined area remains constant. By contrast, a multi-robot system is like a cooperative team; each robot in the system has to update its memory based on the information it receives from the others. This feature, then, increases the complexity of developing coverage algorithms for multiple robots to carry out tasks. And if we think about the motives for building a multi-robot system rather than a single-robot system, the primary reason is to save time. We expect that a team with five robots could work five times as quickly as a single robot to perform a mission under ideal conditions. Based on this line of reasoning, we might build a multi-robot system with this time-saving goal in mind. Indeed, most algorithms for multi-robots have been proved to be time saving, but there is still some room for improvement.

A. PURPOSE OF RESEARCH

In the past, in order to conduct a mine countermeasure (MCM) mission, we risked the lives of personnel searching for mines in the littoral area near the primary channel for warships going in and out. If we could substitute robots for military personnel to execute this kind of missions, we would no longer be necessary to risk human lives. The purpose of his research is to demonstrate a multiple robot coverage algorithm applicable to MCM scenarios.

B. APPROACH

The steps listed below demonstrate how this work is developed gradually. We first do some study on the multi-robot coverage algorithms. Then, we determine an algorithm by interests. Develop the code based on the pseudo code of paper in MATLAB program is the next move. Last, we simulate different experiment by implementing the algorithm.

1. Search for Related Coverage Algorithm

The essence of the fundamental concept of path planning algorithms is well-classified in [1]. From [1], we know coverage could be completed to some extent through the cell decomposition method, which consists of three branches. Most coverage algorithms are developed based on the knowledge and concepts from those branches. The branches within the cell decomposition method are classified as approximate cellular decomposition, exact cellular decomposition, and semi-approximate.

a. Approximate Cellular Decomposition

All the grids of the area to be covered are defined to have same size under approximate cellular decomposition. Using this type of cellular decomposition, the target region, such as the size and shape of obstacle, is presented approximately because the space is composed of uniform grids. A comprehensive concept, which is shown in Figure 1, is demonstrated in [2]. Usually the size of the grid is also the size of the robot footprint. Coverage is complete after robot enters all the grids in the domain field.

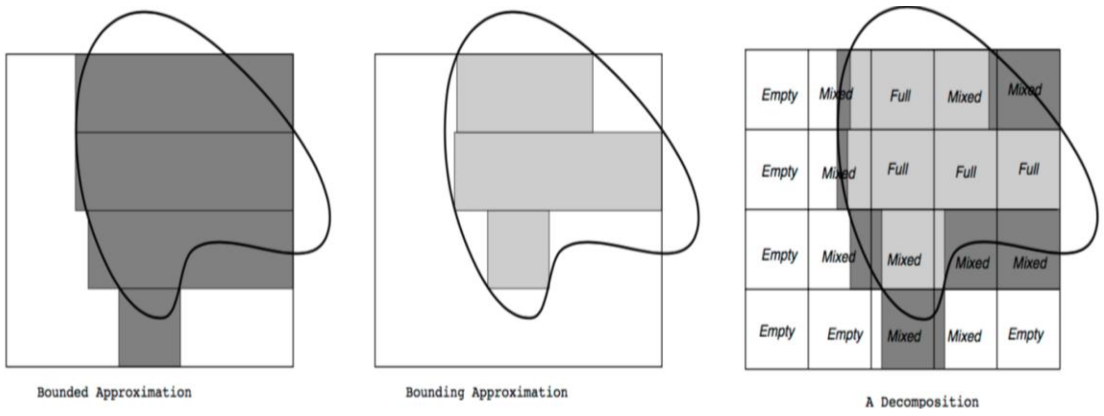


Figure 1. Approximation of an Obstacle by Uniform Grids. Source: [2].

One of popular algorithms to illustrate approximate cellular decomposition is the wavefront algorithm. The wavefront algorithm is an algorithm marking the goal as zero and the adjacent areas around zero are marked as one. Then the adjacent areas around one are marked as two and so forth. Thus, each grid in the domain area is numbered in an ascending order. A specific area marked as any number higher than the goal is defined as a start zone. A robot moves from start point to the goal by following a gradient rule. An efficient path is defined when the sum of numbers on the path is minimized as illustrated in Figure 2.

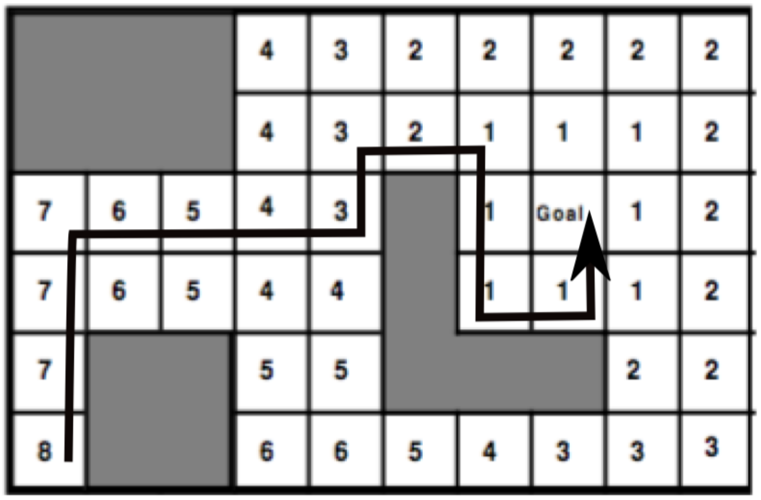


Figure 2. An Efficient Path Found by the Wavefront Algorithm. Adapted from [3].

The best example to illustrate this is found in [3], where Zelinsky et al. implemented a conventional wavefront algorithm to determine a coverage path, as shown in Figure 3.

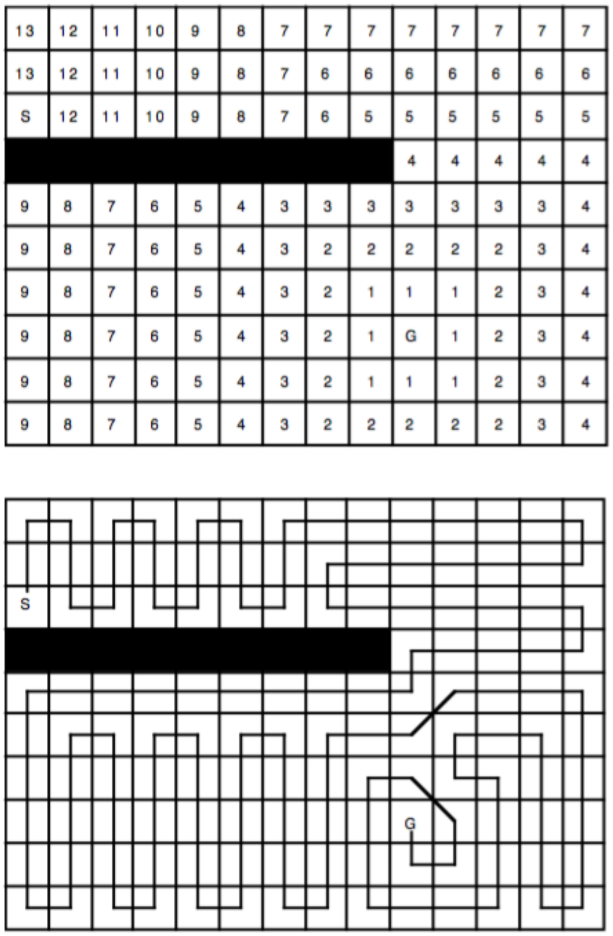
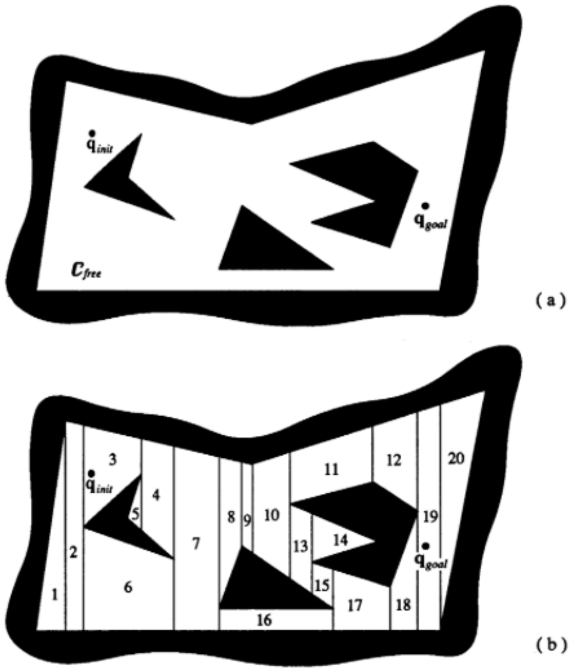


Figure 3. Path of Complete Coverage Executed by Conventional Wavefront Algorithm. Source: [3].

b. Exact Cellular Decomposition

There is no uniform cell in exact cellular decomposition. The target region is exactly defined in the environment. Trapezoidal decomposition in [4] is the best case to convey this concept. Figure 4 illustrates the graphical prototype. As soon as the cells are defined, a robot can implement back-and-forth motion to complete coverage.



In (a) the initial position, goal point, and populated obstacles in the environment are shown. In (b), the free space is decomposing into trapezoidal cells.

Figure 4. Example of Exact Decomposition—Trapezoidal Decomposition. Source: [4].

c. Semi-approximate Decomposition

Semi-approximate cells are cells that are partially discretized in width without uniform height. The robot follows a depth-first order of motion in this method. Thus, the robot moves along the boundary of the domain area until it meets the deepest position of the field. Then, it goes up if it has met the discretized width line or moves horizontally along the boundary. This path will result in some of area not being explored and others might be visited twice. The example illustrated in [1] is shown in Figure 5.

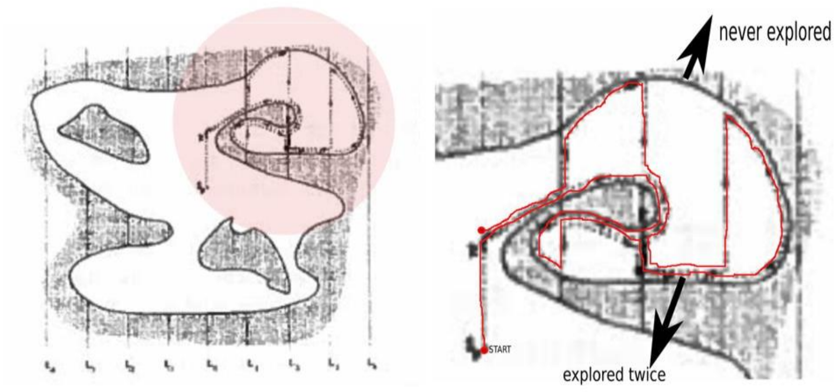


Figure 5. Example of Semi-approximate Decomposition. Adapted from [1].

2. Coverage Algorithm of Interest

While many papers concentrate on coverage completeness and the time it takes, the issue of repeated coverage is less studied. As a result, the algorithm this work extends is known as “Complete Multi-Robot Coverage of Unknown Environments with Minimum Repeated Coverage” [5]. The environment is designed to have exact cellular decompositions. Furthermore, the robots have no prior knowledge of the environment. The pseudo code presented in the paper is performed in MATLAB code. The robots perform behaviors such as obstacle avoidance, teammate avoidance, and boundary avoidance. Other behaviors, such as searching in convex or concave shaped obstacles, are ignored because the purpose is related to conducting an MCM mission in which warships have to keep a specific distance from obstacles. As a result, searching in convex or concave shaped obstacles has no value for our discussion.

3. Develop MATLAB Code from Pseudo Code

An algorithm is established under the topology concept, which is point based, and it is performed as shown in a two-dimensional grid-based implementation. We are looking for an efficient coverage algorithm for a multi-robot system to complete an MCM mission. In such a scenario, the most complex situation to explore is the area distributed with rocks and reefs. We created an environment that emphasized the area around obstacles and observed the performance of robots under the algorithm. In order to implement the algorithm from the pseudo code, the easiest way is to analyze it from the kinematic perspective.

In the work, we use the MATLAB program as our primary platform to establish the environment, the behaviors of each robot, and the way each robot moves to its next position. The main behaviors are boundary, obstacles, and partners avoidance. “Neighborhood” provides a certain distance outside and around the boundary, obstacles, and robots, which guarantees the robots’ avoidance of one another. The algorithm drives each robot to go to the point which has minimum moves to choose after its arrival. When they reach this point, the robots switch modes from normal mode to wrap mode to complete the coverage.

4. Simulation Experiment

MATLAB program is also used as a simulator to evaluate the algorithm using a different number of robots and obstacles. The number of robots ranges from one to three. The area of the domain field is constant. And, the total area containing obstacles is the same. The number of obstacles ranges from one to three as well. Thus, we set area size sixteen for one obstacle environment, size eight for two obstacles environment, and two obstacles of size four along with another size eight obstacle for three obstacles environment. To sum up, there are nine conditions to estimate in the simulation experiment. The number of steps and whether the coverage is completed are essential topics to discuss in Chapter IV.

C. CONTRIBUTIONS

In order to find out if the algorithm is applicable to an MCM mission, we implement it in the previously described environment within a specific domain area containing obstacles. Since all we have is pseudo code for the algorithm, from a very fundamental perspective, we implement the algorithm in a two-dimension grid-based scenario. This work will help people who want to understand coverage algorithms in an MCM mission, and simultaneously, we aim to provide some results to the coverage algorithm field. The following list highlights our intended contributions:

- Analysis of a topology-based, multi-robot coverage algorithm from pseudo code.
- Implementation of a topology-based multi-robot coverage algorithm in MATLAB.
- Extension of a topology-based multi-robot coverage algorithm for application to an MCM scenario.
- Evaluation of the performance of a coverage algorithm in a variety of simulated scenarios.

D. ORGANIZATION OF STUDY

There are five chapters in the study. In Chapter I, we introduce primary structures of this work. Researches related to multi-robot coverage algorithm are studied in Chapter II. After determining an algorithm, the code to implement it is developed and explained in Chapter III. In Chapter IV, we implement the developed code of the algorithm in different scenarios, discuss the results from each experiment and analyze the factors that affect the outcomes. In Chapter V, we determine whether the algorithm is applicable for MCM mission, list benefit and drawbacks of the algorithm and demonstrate future works that we can do if time is available.

II. LITERATURE REVIEW

In order to execute a task of mine detection in specific waters, we have to determine an algorithm for the multi-robot system to perform this task. In such a mission, efficiency and accuracy are our top priorities. The former requires that the robots complete the coverage within of the least amount of time, and the latter requires finding out the location of interest correctly. Moreover, within the category of accuracy exists the probability of detection. An example to illustrate this idea is shown in Table 1.

Table 1. Metrics of Probability of Detection

Aims	Is there a mine?		
Is mine detected?		Yes	No
	Yes	Correct	False Alarm
	No	Missed Detection	Correct

We can glean some potential algorithms that might be applied in the mine detection task. Nevertheless, even though most of the algorithms have been successfully implemented in various simulations and experiments, few have been examined in an underwater environment. Moreover, the contribution of some algorithms could not meet our need to perform efficient and accurate coverage in the task. Therefore, additional research is necessary to apply these theoretical algorithms to the practical challenges of an MCM.

A. SPECTRUM OF ROBOT CONTROL ARCHITECTURES FOR ALGORITHMS

Before selecting an algorithm, the following are some basic concepts to differentiate numerous approaches. Perhaps one way to classify or differentiate between these different approaches is based on “behavior-based” versus “system theory based” approaches to the problem.

1. Behavior–Based Approach

Most behavior-based systems are reactive systems, bioinspired, or ad-hoc approaches, which do not require explicit mathematical models of the subsystems or environment. The robot uses the gleaned information to gradually correct its actions depending on the changes in the immediate environment. Behavior-based robots (BBR) are able to perform complex behaviors; however, they have difficulty proving or guaranteeing both stability and performance.

2. System Theory–Based Approach

Neither planner based nor deliberative approaches can verify sensory information or generate motions directly; they require a centralized computer as a medium to do this. The planner could apply the information in the model to generate the most proper sequence of actions for the robot. Thus, the quality of the performance of the robot could be evaluated through those explicit formulas. As a result, system theory based approaches can prove or guarantee both the stability and performance [6], e.g., decentralized control.

Some documents classify the behavior-based design as one of the members of a reactive control system [7], while others suggest that it is much more similar to the hybrid control system, which is in-between the reactive and deliberative control systems [6]. In this thesis, the latter is adopted. Figure 6 illustrates this spectrum of potential approaches, and Table 2 lists the characteristics of each of the different approaches.

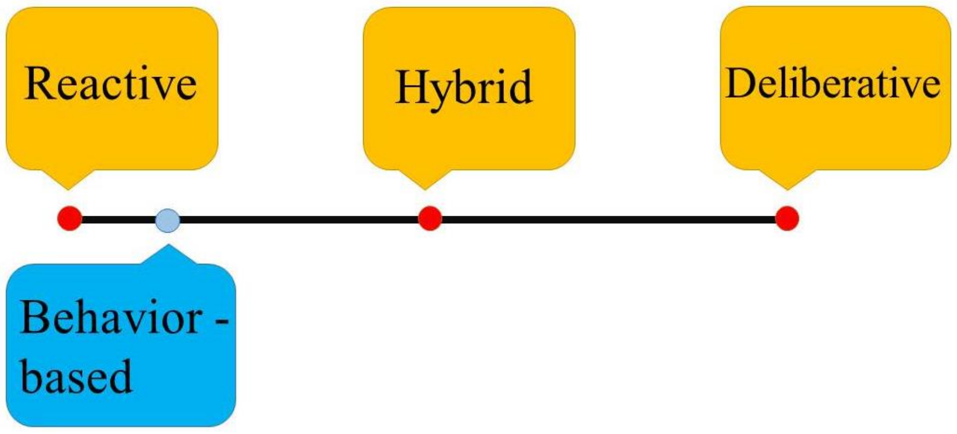


Figure 6. Spectrum of Robot Control Architectures

Table 2. Features of Robot Control Architectures. Adapted from [7], [8].

Features	Reactive	Behavior-based	Hybrid	Deliberative
World model			v	v
Requires large amounts of accurate information			v	v
Flexible for increasing complexity			v	v
Built-in or learned from loading in the past	v	v	v	
Making maps (Design path)			v	v
Avoiding obstacles	v	v	v	
Monitor performance			v	v
Low memory processing	v	v	v	
Internal representaions			v	v
Ability to learn			v	v
Respond quick	v	v	v	
Require three-layer systems (Reactive layer, deliberative layer and intermediate layer between the two)			v	
A sensing system (translate raw sensor input into a world model)				v
A planning system (take the world model and a goal and generate a plan to achieve the goal)				v
An execution system (take the plan and generate the actions it prescribes)				v
Localize a robot relative to a world model			v	v
Decomposition into contextually meaningful units		v	v	
Combine the responsiveness, robustness and flexibility of purely reactive systems with more traditional deliberative methods.			v	
Thinking performed through a network of behaviors	v	v		
Uses distributed representations	v	v		
Responds in real-time	v	v		
Allows for a variety of behavior coordination mechanisms	v	v		

B. STATE OF THE ART

The coverage of single-robot systems has been analyzed for decades. The comprehensive categories of the levels required to complete the coverage are well organized in [1]. For example, we could apply heuristic and randomized approaches to develop an algorithm for vacuuming/sweeping robots because such applications do not have to take time into consideration. As a result, if given long enough time, a robot will complete its task as accurately as possible. However, in the demining problem, both efficiency (minimum time) and completed coverage (accuracy) are required. Therefore, some advanced algorithms based on exact cellular decomposition for multi-robot systems are developed.

The algorithm proposed in [9] is based on boustrophedon decomposition [10], which is a kind of exact cellular decomposition. Choset and Pignon defined that a cellular decomposition is the construction of a graph in the desired environment, which in turn could be applied to plan coverage. The authors further wrote, “Once the specific cell has been visited, the robot uses the structure of the graph to plan a path to an uncovered area, and as the graph has no unexplored places, the coverage is complete” [10].

The square robots in the algorithm are equipped only with intrinsic contact sensors to detect the boundary of their finite environment, and to perform a complete and efficient coverage in a finite environment. This algorithm is called DC_R, which includes three distinct elements: the CC_{RM}, feature handler, and overseer. With tasks properly designed by DC_R, those three components could execute coverage equally capably and efficiently in shared, connected, and rectilinear settings. The first, $CC_{RM,}$ is in charge of covering the environment by constructing the cellular decomposition increasingly. The knowledge assigned to CC_{RM} are cellular decomposition C ($C = \{C_0, \ldots, C_n\}$) and the current position of robot only. At this stage, each robot does not have to take other robots into consideration. The feature handler is the second part of DC_R, except for deriving pre-specified kinds of features from C, it is also responsible for communications between robots and determination of the relative location of the several robots. Finally, the last part, overseer, deals with the incoming data from robots and modifies the coverage of C; in other words, the overseer incorporates the data of a new unexplored area with the data from the original one and updates the new coverage for CC_{RM}. The control concept of the algorithm demonstrated here could be viewed as a centralized perspective, because there is a decision maker (feature handler) who is in charge of communication between robots.

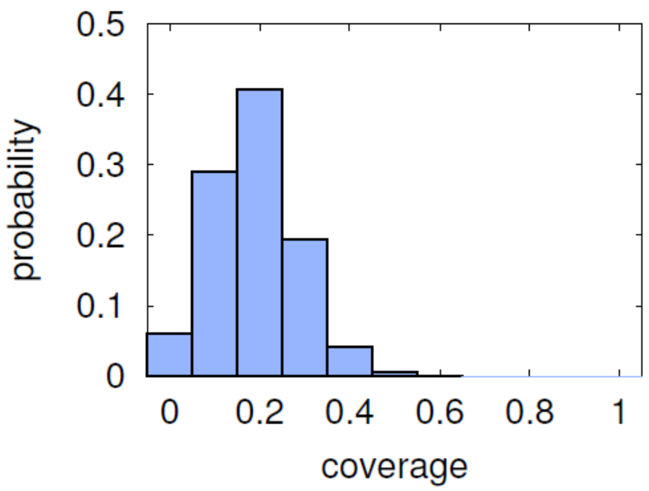
To sum up, the DC_R was derived from [10], which is the complete coverage based on a single sensor-based robot. By applying multiple robots and the DC_R algorithm in the same environment as [10], DC_R has successfully proved that a multi-robot system, whose efficiency outperforms the one in [10], could complete coverage in a rectilinear environment. The algorithm, however, could not guarantee both the minimum time and repeated path required for the coverage.

Apart from other algorithm methods of that apply the cellular decomposition approaches, [11] applies a generated map to solve the problem of coverage. Stachniss and Burgard suggest that "generating maps is one of the fundamental tasks of mobile robots" [11]. The authors further assert, "Some of the most important aspects are the localization of the vehicle during mapping, appropriate models of the environment and the sensors, as well as strategies for guiding the vehicle" [11]. Among the many algorithms for the coverage of multi-robot systems, efficiency is a primary goal, and it is appropriately defined as the minimum time required for completing coverage. The sensor might have some influences on efficiency since there might be noisy signal in communication when the distance between the robots and the control center becomes larger. To be efficient, the paper concentrates on handling the problems of localization and mapping simultaneously; meanwhile, the team also develops an approach for mobile robots to learn accurate maps from noisy range sensors [11].

The definition of the task of exploration here is to guide a vehicle in such a way that it applies the sensor to sweep the environment. In other words, exploration maps the unknown area so as to determine the conditions and the characteristics of the environment. While coverage is the way that a robot travels within a given map when in search of an interest, the authors agree that a common strategy to perform exploration is to extract frontier between known and unknown places and to visit the closest unexplored area [11]. Yet, they also reveal that the approaches previously mentioned only recognize scanned and unscanned fields. They do not take the actual information collected at each viewpoint into consideration [11].

To avoid the uncertainty of the robots in the map, they install a posterior for every cell. As a result, the team could monitor the change of entropy locally. The authors define entropy as a general measure for the uncertainty of a belief [11]. The values of entropy extend from 0 to 1, where 0 means well-explored area and 1 means unexplored area.

The team further introduces the underlying concepts for their algorithm. First, the introduction of coverage maps, which ranges also from 0 to 1, where 0 stands for empty area, and 1 represents occupied fully. The figure between 0 and 1 depends on the portion of the obstacle in a grid cell. The result is shown in the histogram provided in Figure 7. Secondly, the way to update coverage maps is introduced. The team assumes their sensors would provide the information of distance. By applying the Bayes rule, they are able to convert the distance information to coverage values. The following strategies, in our point of view, Closest Location (CL), Maximum Information Gain (IG) and (IG-WIN), and Combination of IG and CL (IG-CL) are the essence of their algorithm.



The image shows a cell visited by a robot for measurement 30 times. Among 30 measurements, the highest probability of an obstacle in this cell is about 0.2, which means 20% of the area is covered by an obstacle.

Figure 7. The Result of Probability of Coverage in Histogram. Source: [11].

The strategy of Closest Location is to run the robot to the nearest location, where it is able to collect the information from the grid cell that has not been examined enough. The way to determine the uncertainty is based on the measurements of entropy. It would be lower than a specific value or does not have significant change (below 0.001) for a well-explored area. The CL considers only the shortest distance calculated from robot's current position.

From a specific viewpoint, the information is gained based on the measurement of the change of entropy, which is introduced by incorporating a measured distance. In the strategy of Maximum Information Gain, authors further integrate the distance into a new cell. Because the information of the measurement obtained by the robots is the sum of the information of whole cells, we do not know the exact data from the sensor at a specific location. Thus, all possible measurements are integrated to evaluate the expected information gained at a certain point. Then we have our next viewpoint. However, the position of next viewpoint is located at the original coordinate, which does not consider the location of robot. The strategy of IG-WIN is proposed to improve this disadvantage by providing the robot with the distance from it to the next viewpoint.

The final strategy is the combination of IG and CL. To find an ideal tradeoff performance for the robot, the team develops a formula to define the behavior of that robot in which the result would be between 0 and 1. If the result is zero, the robot would perform the behavior of CL; otherwise, it performs the behavior of IG. This kind of algorithm could be classified as a system theory based approach since the change in the coverage map is based on the Bayes rule.

The algorithm is devoted to creating a system that can generate more accurate maps of cells. The authors state that the algorithm decides the best trade-off between the number of necessary measurements and calculates the length of the resulting paths [11]. However, the algorithm does not guarantee the required minimum time.

The approach described in [12] tries to solve the coverage problem using sensor-based robots. First, Batalin and Sukhatme propose a hypothesis with a scenario in which the environment is unknown and the Global Positioning System is unavailable. Secondly, they propose two algorithms to perform the coverage task successfully with the help of only local sensing and local interaction between robots. Thus, these algorithms employ the Informative Technique and the Molecular Technique, respectively. The robots using the former method could be viewed as centralized, since the robots make decisions after analyzing the measurement from other robots. The latter, conversely, is based on the decentralized concept, because each robot makes its own decision depending on its own measurement.

Each robot is equipped with two planar laser range finders with a 180-degree field of view, color camera, vision beacons, and wireless communicator. In their system, the robots are not only able to perform obstacle avoidance but also of to be mutually repelled by one another within a certain distance limited by the sensors. The experiment is carried out according to three approaches. The first is called the Informative, which is based on the idea of assigning local identities to the robots temporarily as they are within sensor range of each other. Then the information mutual to the relative positions of each robot would be shared by the wireless network. These two stages help the robots spread out in a coordinated manner. The second approach is named Molecular [12], the underlying concept of which is simpler than the Informative. Without the local identification and communication among the robots, the task for each robot is only to move in a chosen direction away from the others. The perception of the robot to distinguish itself from another one is required in both approaches. The third approach is termed Basic, in which the robots only have the ability of to avoid obstacles. The robot does not have the ability to distinguish itself from the others. In the experiment, the results of the Informative and Molecular approaches outperform the Basic approach, because the standard deviations of the Basic approach increase when the robots are positioned in different locations and as the number of robots changes. Both the Informative and the Molecular approaches show small standard deviations when the robots are located at distinct positions and when the number of robots changes. However, the Molecular approach performs a bit better than Informative approach because in the Informative approach the robot has to stop to identify other robots within the range of its sensors. This task is costly in terms of speed and converge time. In the Molecular approach the robot only has to find the best path away from other robots, which makes this approach the most efficient one. The team plans to modify the Informative approach by shortening the time a robot has to stop to identify other robots.

The idea of applying a multi-robot distributed coverage algorithm, which is proposed in [13], manages to solve the humanitarian de-mining problem. With the implementation of Boustrophedon decomposition and the Cycle algorithm, Acar and Choset point out that “no robot remains idle while there are areas to be covered” [13]. In other words, their coverage is efficient. Three phases are introduced in the Cycle algorithm. They are the forward phase, reverse phase, and closing phase. Three scenarios are envisaged as environments that the robots might encounter. The first is the Cell with no obstacles. In this setting, the robot performs the forward and reverse phases. The robot confirms that the current cell coordinates are correct and renews the coverage map. Furthermore, the robot is able to assume that the neighbor cell, which is unexplored, has the exact same size as the present cell and to add it into the adjacency graph. In the second scenario, termed Cell with obstacles, which entails blocking part of the cell’s width, the robot uses the sensor to detect and memorize the obstructing position. The recognized location would be viewed as a starting point for the adjacent cell to be added later. The third scenario is named Cell divided obstacle. While performing forward and reverse phases, the robot meets an obstacle blocking the path. Thus, it updates the size of the cell and assumes that the adjacent cell has the same height as the current one. The outline of the graph would be rectified once the robot detects the scenario of the Cell with obstacles blocking part of the cell’s width. Communication plays an important role in this algorithm, because the robot will broadcast the new information to the team after it finishes the coverage. After receiving the message, each robot combines the new information with its own picture. Lack of communication will lead to overlap coverage. This paper [13] poses an algorithmic approach to the distributed complete coverage problem, which is quite successful because there is no idle behavior when a robot completes its own coverage. However, the authors reveal that the “bandwidth limitations and congestion in narrow areas are potential limiting factors” [13].

The aim of [5] is to apply a decentralized algorithm to complete coverage in an unknown environment. Meanwhile, instead of looking for efficiency, they focused on how to minimize repeated coverage, which indirectly increased the efficiency of coverage. There are two modes in their algorithm: normal mode and wrap mode. Under these modes, onboard sensors and a network for both communication and sharing information among the robots are required. The robots treat explored areas as spurious obstacles. The diameter of the circular footprint of each robot is defined as d_r. In the perception of a robot, there are regions that need to be covered, the actual subspace of a region that could be reached by robots, uncovered space, already covered space, spaces between obstacles that are less than $2d_r$ apart, boundaries, real obstacles, spurious obstacles, and other robots. Under normal mode, a robot looks for a real obstacle and a spurious one while it checks its internal path simultaneously. Then the robot searches for a nearest point, which allows the robot to keep the distance $4d_r$ and $2d_r$ from the real obstacle and spurious obstacle, respectively. The wrap mode is activated when such a point cannot be located. At that moment, a robot travels along its explored area in a counterclockwise direction from its current position and looks for any free space. If it does not reach a free point by the time it arrives at the original point, the robot then patrols along its explored area in a clockwise direction. Under this procedure, if there is an unexplored area that had initially been ignored, the robot shifts from wrap mode to normal mode. Once an obstacle opening is detected, it covers the area by back-and-forth motion; meanwhile, it checks whether the distance is $4d_r$ or $2d_r$ from the real obstacle and spurious obstacle, respectively. When the condition is true, normal mode takes the place. The repeated coverage only happens to the areas where the distance between obstacles is less than $2d_r$. As a result, no unnecessary repeated coverage occurs under this general scenario, which further improves the general efficiency of the study. The team has successfully minimized the repeated coverage through simulations.

The main idea of [14] is to focus on an online multi-robot coverage algorithm based on approximate cellular decomposition. They introduced Multi-Robot Spanning Tree Coverage (MTSC). By implementing the algorithm, they expected their coverage to be complete, non-redundant, and robust. The definition of complete is that all the paths are covered by robots. And, the criteria for non-redundant is that the same place is not scanned more than twice. Robustness ensures that as long as a robot is still alive, the coverage will be finished. The robots are able to sense, determine, communicate, and cooperate with peers while exploring. The environment is unknown. In the algorithm, all robots are given an absolute location. The team conducted several trials with real-world robots. The authors claim "The algorithm works well in diverse environments and group sizes" [14]. Nevertheless, because a robot can make a turn by 90 degrees once each time, it limits its efficiency for coverage.

A novel approach called the multiple aspect coverage (MAC) algorithm has been derived in [15] to plan paths for detecting clusters of interests. Two requirements for this algorithm are the information about target location and well-operated sensors for an autonomous underwater vehicle (AUV). The AUV will operate straight-line mode to approach the targets by three swaths. The algorithm calculates proper angle, the width of swath, and the fewest turns required to cover the clusters of targets. In an experiment involving three targets, the path planner of MAC uses less time, less distance, and fewer turns to localize the position of the targets; thus, it outperforms the other two approaches (Standard and Human operator). The MAC algorithm could be viewed as a budding star in the MCM field, even though the experiment only applied one AUV. Table 3 and Figure 8 contain some experiment results from [15].

Table 3. Performance Metrics for the Three-target Case. Source: [15].

Path Planner	**Time (min)**	**Distance (m)**	**Turns**	**Probability**
MAC	12.65	1461.1	8	0.952
Standard	24.64	2699.5	17	0.959
Human operator	12.13	1317.6	8	0.759

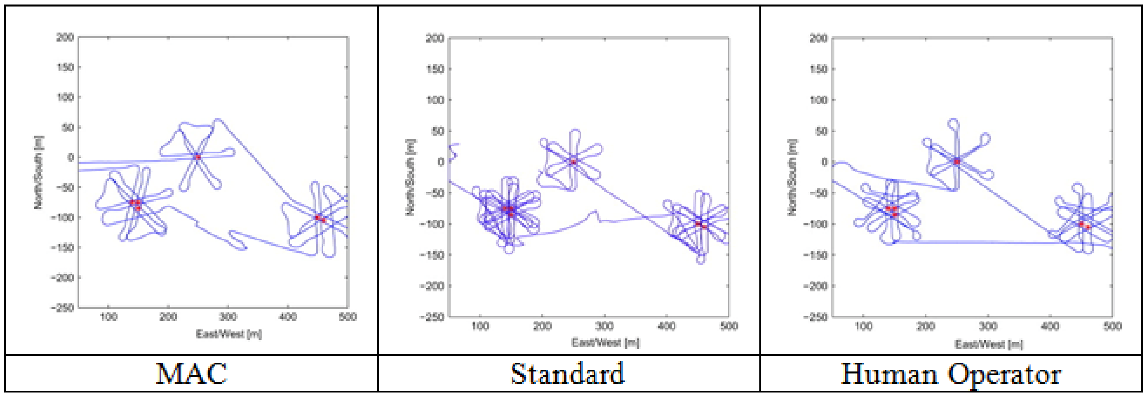


Figure 8. Experiment Result for the Three-Target Case. Source: [15].

C. ALGORITHMS SUMMARY

Table 4 lists studies related to the classification of algorithms based on types of control, characteristics, number of robots, interaction among robots, assumptions, and the contents of the studies.

Table 4. Classification Determined by Assorted Characteristics

Papers	algorithms	Type of Control			Characteristics			Number of Robots		Interaction among Robots		Assumptions	Contents
		Reactive	Deliberative	Hybrid	Performace	Stability	Efficiency	single	multi-	Centralized	Decentiralized		
2000 (Complete Distributed Coverage of Rectilinear Environments)	CC_R		v		v	v	Doesn't prove		v		v	Decomposition Navigation Check its own location	Introduce algorithm Proof Simulation
	Feature handler		v		v	v	Doesn't prove		v	v		Communication	
	Overseer		v		v	v	Doesn't prove		v		v	Modified Cellular decomposition	
2003 (Exploring Unknown Environments with Mobile Robots using Coverage Maps)	Coverage Maps		v		v	v	Doesn't prove		v		v	Navigation Scan	Introduce algorithm Experiment
2002 (Spreading Out: A Local Approach to Multi-robot Coverage)	Informative			v	v	v	v		v	v		Naviagation Communication Determine Obstacle-avoidance	Simulation
	Molecular			v	v	v	v		v		v	Naviagation Determine Obstacle-avoidance	
	Basic	v							v		v	Obstacle-avoidance Navigation	
2006 (Distributed Coverage with Multi-Robot System)	Cycle			v	v	v	Doesn't prove		v	v		Navigation Communication	Introduce algorithm Simulation
2005 (Complete Multi-Robot Coverage of Unknown Environments with Minimum Repeated Coverage)	Normal		v		v	v	v		v	v		Scan and Determine Communication Turn its own path into spurious obsatcles	Introduce algorithm Analysis Proof Simulation
	Wrap		v		v	v	v		v	v		Scan and Determine Communication	

Papers	algorithms	Type of Control			Characteristics			Number of Robots		Interaction among Robots		Assumptions	Contents
		Reactive	Deliberative	Hybrid	Performace	Stability	Efficiency	single	multi-	Centralized	Decentiralized		
2006 (Towards Robust On-line Multi-Robot Coverage)	On-line MSTC		v		v	v	Doesn't prove		v		v	Decomposition Navigation Check its own location	Introduce algorithm Proof Simulation
	ORMSTC			v	v	v	Doesn't prove		v	v		Sense and Determine Communication Cooperative	
2012 (Theory and experimental results for the multiple aspect coverage problem)	MAC		v		v	v	v	v			v	Travel in straight line Sense and Detect	Introduce algorithm Experiment Proof Simulation

Table 5 Classification determined by assorted characteristics

D. DETERMINE ALGORITHMS FOR THE MCM TASK

Even though we have listed the characteristics for every algorithm, it is difficult to determine which algorithm is the most suitable for the MCM task. The approach in [9] could develop complete coverage for a rectilinear environment with a multi-robot system, but it does not consider the problem of avoiding obstacles, which is a required behavior in our mission. The work of [11] introduces an algorithm based on the coverage map, and it could be modified based on the sensory information. However, the algorithm focuses on finding the exact boundaries of each cell instead of the position of the targets. Thus, this algorithm is not applicable in an MCM mission. In [12] three methods to perform the coverage are demonstrated. Among these methods, the Informative algorithm has the potential to be used in our mission since it has ability to recognize other robots in the predefined environment; however, it takes too much time when identifying peers. As a result, in consideration of efficiency, the Molecular approach might be a possible solution. Based on the exact cellar decomposition algorithm, [13] could complete the coverage and also perform the behavior of avoiding obstacles. Thus, it might be a potential candidate as well. In [5], it is worthwhile to think about an algorithm that calculates the minimum repeated coverage within a predefined territory. The set environment of the experiment has obstacles of various shapes, and the result shows this algorithm performs efficiently and sweeps the coverage area completely. The algorithm applied in [14] is based on the approximate cellar decomposition method. Even though it executes well for demonstrating a complete coverage in the experiment and uses less time compared with the exact cellar decomposition method, the accuracy of the position of targets does not need to be determined in the experiment. The novel approach in [15] is a state-of-the-art technique useful for our mission, even though it uses only one AUV in its experiment. The mine should be distributed on the bottom of the ocean in a cluster formation. Thus, the MAC algorithm has been proven to be the best way to locate the position of the mines.

III. ALGORITHM DEVELOPMENT

A. GENERAL TOPOLOGY–BASED ALGORITHM

The algorithm presented in this chapter, including the equations and notation associated with the algorithm, is proposed in [5]. A contribution of this thesis is the interpretation of the algorithm and its implementation in a grid-based environment. The reason why it is called the minimum repeated coverage algorithm is that when the robot cannot find another route to explore, it moves along the boundary of its own explored area to search for a new route. Following this this behavior, it only repeats the route of the boundary of its own covered area instead of other routes chosen by chance. There are two modes under this algorithm: normal mode and wrap mode. Table 5 summarizes the basic nomenclature for the algorithm. Table 6 lists the meaning of the symbols used in the following figures.

Table 5. Nomenclature. Source: [5].

q_{ri}	Current position of robot r_i;
d_r	Diameter of circular footprint of each robot;
$D_c(q_{ri})$	A disc centered at q_{ri} with radius 0.5 d_r;
E	Region in $\mathbb{R}^2$ that the robots need to cover;
E_{ex}	Actual subspace of E that can be reached by robots;
E_{ux}	Uncovered space in E_{ex};
$E_{c,i}$	Space already covered by robot r_i;
E_{nrw}	Spaces between obstacles that are less than 2 d_r apart;
O_k	Obstacle k;
n_r	Number of robots in the team;
n_{ob}	Number of obstacles that the robots know of;
$B(X)$	Boundary of a space X;
$N(X,\delta)$	δ-neighborhood of the space X;
$\widehat{A}_{a_1a_2}$	An arc, A from a_1 to a_2 in the anticlockwise direction.

Table 6. Meaning of Symbols

🔴	Current position of qr,i
🟡	Explored points of qr,i on it's boundary
⚫	Boundary points of domain area
🟦	Explored point of qr,i that is not on boundary
○	Explored points of other robot
🟢	qB,i. (From unexplored to explored)
★	Star point, which is used to direct to next point
🔵	Neighborhood of qr,i
🟣	Critical point of qr,i
🟣	Previous point of qr,i
●	Desired next point for qr,i

B. NORMAL MODE

The normal mode is the mode in which robots can move as they keep a certain distance from the domain area, obstacles, and other robots. In addition, the robots must go to the spaces that are bordered by the most covered areas. Since the areas bordered by less covered areas are distinct from those covered by the most covered area, the spaces bordered by the most covered areas have a greater chance of being ignored. This is true even when robots are in wrap mode if they go to the spaces bordered by a less covered area first. Figure 9 shows the pseudo code of normal mode.

```
Normal Mode
1. // Find all possible regions that $r_i$ can move to. Keeping $d_{ob}$ from obstacles and
   $B_E$, and 2 $d_{ob}$ from $E_{c,j}$.
2. Let $B_{N*} = B\left(N\left(E_{c,i}, 0.5d_r\right) \cap N\left(q_{ri}, 0.5d_r\right)\right) - N\left(B_E, d_{ob}\right) - N\left(O_k, d_{ob}\right) -$
   $N\left(E_{c',j}, 2d_{ob}\right)$, for all obstacles within range, and $E_{c,'j}$ are the spaces
   in $N\left(D_c\left(q_{ri}\right), 5d_r\right) \cap E_{c*}$ that do not contain $q_{ri}$.
3. // Classify the regions that should be avoided into $E_{um}$.
4. Let $E_{um,i} = E_{um,i} \cup \left(N\left(q_{ri}, d_{ob}\right) \cap N\left(O_k, d_{ob}\right)\right) \cup \left(N\left(q_{ri}, 2d_{ob}\right) \cap N\left(E_{c,j}, 2d_{ob}\right)\right)$
5. if $\left(B_{N*} = \varnothing\right)$ then
6. // Change mode if there are no permissible paths for expansion.
7. Enter: Wrap mode.
8. else
9. // If permissible paths exist, move to the point bordered by the most $E_{c,i}$.
10. Let $q_n = \arg\max_{q_{n*} \in B_{N2*}} \left(D\left(q_{n*}\right) \cap E_{c,i}\right)$
11. // Ensure that $E_{ux}$ does not become disconnected.
12. if $\left(E_{ux} - D_c\left(q_n\right) = \text{'disconnected'}\right)$ then
13. $B_{N*} = B_{N*} - D_c\left(q_n\right)$
14. Enter: Wrap mode.
15. else
16. Move to $q_n$. Set $E_{c,i} = E_{c,i} \cap D_c\left(q_n\right)$, and $E_{c*,i} = E_{c*,i} \cap D_c\left(q_n\right)$.
```

Figure 9. Pseudo Code of Normal Mode. Source: [5].

1. Description of the Basic Concept of Normal Mode

The unexplored area, $\mathbf{E}_{ux}$ in [5], shall be visited by the robot, which is given as

$$\mathrm{E}_{ux} = \mathrm{E} - \mathrm{E}_c - \bigcup_{k=1}^{n_{ob}} O_k \ , \tag{1}$$

where $\mathbf{E}$ is the domain area, $\mathbf{E}_c$ is the explored area of the robots, and $\mathbf{O}_k$ stands for an obstacle. The equation states that uncovered area is the area that remains after removing explored area and the area occupied by obstacles from the domain area.

The area explored by robots, $\mathbf{E}_c$ in [5], is the sum of the area explored by other robots that does not include the robot itself and is defined as

$$\mathrm{E}_c = \bigcup_{i=0}^{n_r-1} \mathrm{E}_{c,\,\mathrm{i}} \ , \tag{2}$$

where $\mathbf{E}_{c,i}$ is explored area covered by unit robot.

The combination of robot footprints, $\mathbf{E}_{c*}$ in [5], excluding the robot itself, is defined as

$$\mathrm{E}_{c*} = \bigcup_{j=0}^{n_r-1} D_c\left(q_{rj}\right), \tag{3}$$

where $\mathrm{D_c}\left(\mathrm{q_{rj}}\right)$ is the footprint of a unit robot.

Let $\mathbf{E}_{c',j}$ in [5] be the points that intersect the neighborhood made by the robot footprint with length 5 $\mathbf{d}_r$ and the combination of robot footprints. This is can be expressed as

$$\mathrm{E}_{\mathrm{c',j}} = \mathrm{N}\left(\mathrm{D}_c\left(q_{ri}\right), 5d_r\right) \cap \mathrm{E}_{c*} \ , \tag{4}$$

where $\mathrm{D}_c\left(\mathrm{q}_{ri}\right)$ is a disc centered a $\mathbf{q}_{ri}$ with radius $0.5\,\mathbf{d}_r$ and $\mathbf{d}_r$ is the diameter of the circular footprint of each robot. Figure 10 shows how $\mathbf{E}_{c',j}$ could be found, and Figure 11 displays the scope of the neighborhood of $\mathbf{E}_{c',j}$.

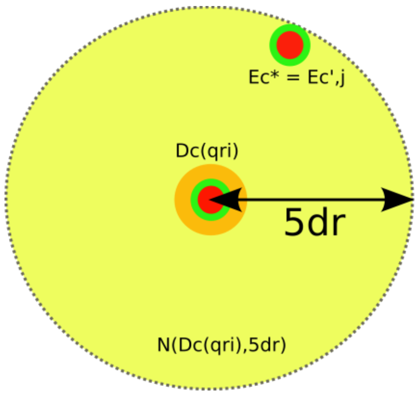


Figure 10. The Basic Concept of Defining $\mathbf{E}_{c',j}$

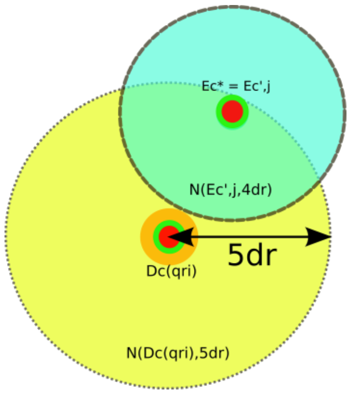


Figure 11. Illustration of $\mathrm{N}\left(E_{c',\mathrm{j}},\ 4\mathrm{d}_r\right)$

Let $\mathbf{B}_{N*}$ in [5] be an arc of the boundary that a robot can move to, which is defined as

$$\mathrm{B}_{N*}=\mathrm{B}\Big(N\big(E_{c,\mathrm{i}},\ 0.5\mathrm{d}_r\big)\cap N\big(q_{ri},\ 0.5\mathrm{d}_r\big)\Big)\text{-}\mathrm{N}\big(B_E,\ \mathrm{d}_{ob}\big)\text{-}\mathrm{N}\big(O_k,\ \mathrm{d}_{ob}\big)\text{-}\ \mathrm{N}\big(E_{c',\mathrm{j}},\ 2\mathrm{d}_{ob}\big)\,, \tag{5}$$

where $\mathbf{B(N(E}_{c,i}\mathbf{,}\ 0.5\mathbf{d}_r\mathbf{)}\cap\mathbf{N(q}_{ri}\mathbf{,}\ 0.5\mathbf{d}_r\mathbf{))}$ is the boundary of the intersection of the neighborhood of $\mathbf{E}_{c,i}$ with $0.5\mathbf{d}_r$, and the neighborhood of $\mathbf{q}_{ri}$ with $0.5\mathbf{d}_r$. $\mathbf{N(B}_E\mathbf{,d}_{ob}\mathbf{)}$ is the neighborhood of $\mathbf{B}_E$ with $\mathbf{d}_{ob}$, where $\mathbf{d}_{ob}$ in [5] is given as

$$\mathbf{d}_{ob} = 2\mathbf{d}_r\,. \tag{6}$$

$\mathbf{N(O}_k\mathbf{,d}_{ob}\mathbf{)}$ is the neighborhood of $\mathbf{O}_k$ with $\mathbf{d}_{ob}$, and $\mathbf{N(E}_{c',j}\mathbf{,}2\mathbf{d}_{ob}\mathbf{)}$ is the neighborhood of $\mathbf{E}_{c',j}$ with $2\mathbf{d}_{ob}$. Figure 12 demonstrates the limited direction in which a robot could move. Figure 13 shows the general concept how a robot moves from its current position to the next position.

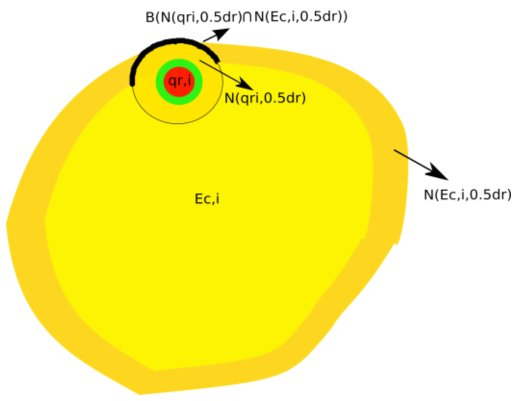


Figure 12. Illustration of Boundary of Intersection of $\mathbf{N}(\mathbf{q}_{ri}, 0.5\mathbf{d}_r)$ and $\mathbf{N}(\mathbf{E}_{c,i}, 0.5\mathbf{d}_r)$

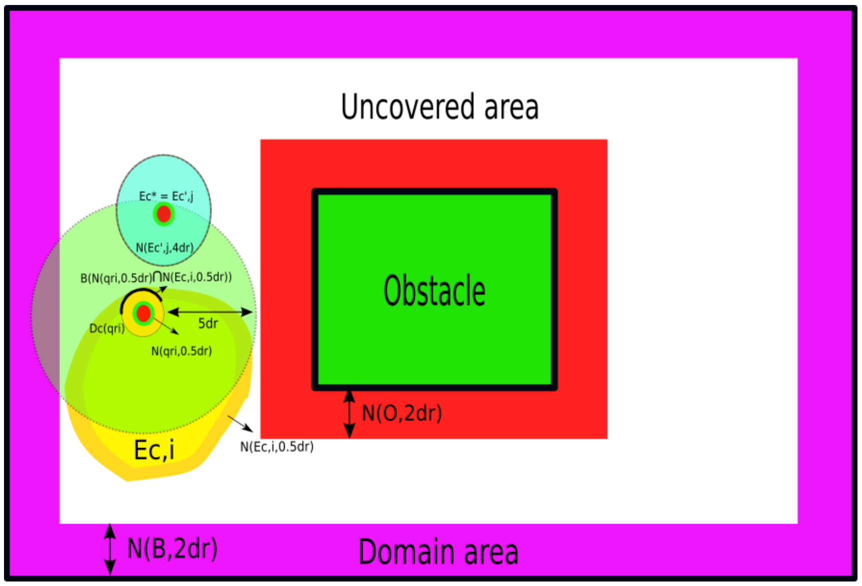


Figure 13. Illustration of $\mathbf{B}_{N*}$

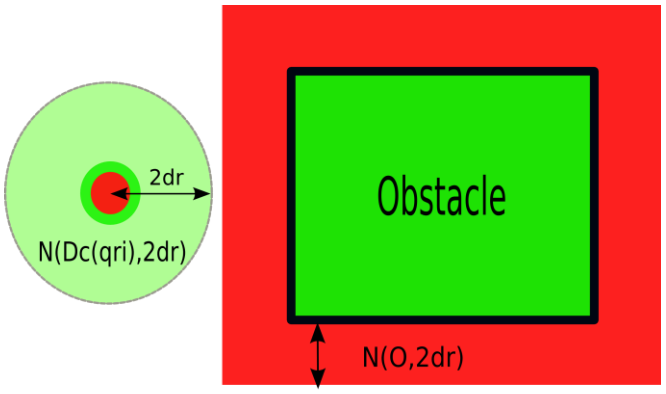
Let $\mathbf{E}_{um,i}$ in [5] be the subset of $\mathbf{E}_{ux}$ and an empty space initially for each robot. $\mathbf{E}_{um,i}$ is described as

$$\mathrm{E}_{um,i} \subseteq \mathrm{E}_{ux} \text{ be } \varnothing \,. \tag{7}$$

Let $\mathbf{E}_{um,i}$ in [5] be defined as the equation when operating the algorithm

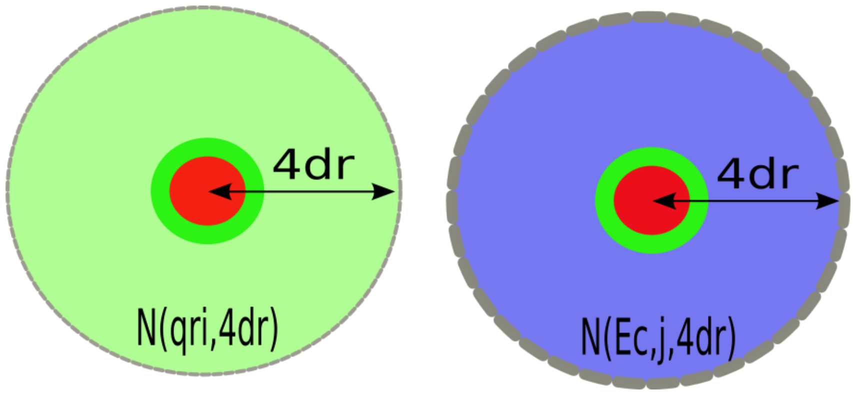
$$\mathrm{E}_{um,i} = \mathrm{E}_{um,i} \cup \left(\mathrm{N}(q_{ri}, d_{ob}) \cap N(O_k, d_{ob})\right) \cup \left(\mathrm{N}(q_{ri}, 2d_{ob}) \cap N\left(\mathrm{E}_{c,j}, 2d_{ob}\right)\right), \tag{8}$$

where $\mathbf{N}(\mathbf{q}_{ri}, \mathbf{d}_{ob})$ is the neighborhood of $\mathbf{q}_{ri}$ with $\mathbf{d}_{ob}$ and $\mathbf{N}(\mathbf{O}_k, \mathbf{d}_{ob})$ is the neighborhood of $\mathbf{q}_{ri}$ with $\mathbf{d}_{ob}$. $\mathbf{N}(\mathbf{q}_{ri}, 2\mathbf{d}_{ob})$ is the neighborhood of $\mathbf{q}_{ri}$ with $2\mathbf{d}_{ob}$ and $\mathbf{N}\left(\mathbf{E}_{c',j}, 2\mathbf{d}_{ob}\right)$ is the neighborhood of $\mathbf{E}_{c',j}$ with $2\mathbf{d}_{ob}$. The intersection of $\mathbf{N}(\mathbf{q}_{ri}, \mathbf{d}_{ob})$ with $\mathbf{N}(\mathbf{O}_k, \mathbf{d}_{ob})$ and the intersection of $\mathbf{N}(\mathbf{q}_{ri}, 2\mathbf{d}_{ob})$ and $\mathbf{N}\left(\mathbf{E}_{c',j}, 2\mathbf{d}_{ob}\right)$ must be empty sets to keep $\mathbf{E}_{um,i}$ empty. Figure 14 and Figure 15, respectively, convey these ideas graphically.



The intersection of $\mathbf{N}(\mathbf{q}_{ri}, \mathbf{d}_{ob})$ and $\mathbf{N}(\mathbf{O}_k, \mathbf{d}_{ob})$ should always be an empty set, where $\mathbf{d}_{ob}$ equals $2\mathbf{d}_r$.

Figure 14. Illustration of Intersection of $\mathbf{N}(\mathbf{q}_{ri}, \mathbf{d}_{ob})$ and $\mathbf{N}(\mathbf{O}_k, \mathbf{d}_{ob})$



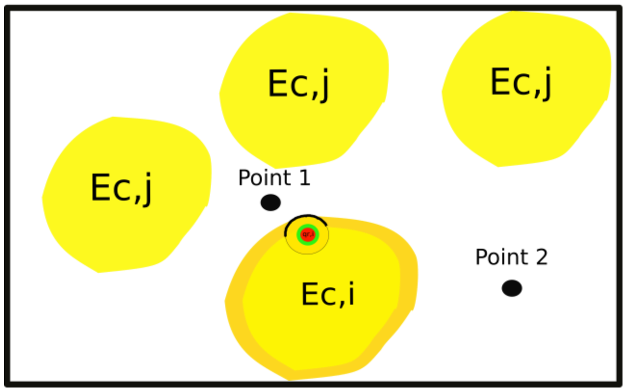
The intersection of $\mathbf{N}(\mathbf{q}_{ri}, 2\mathbf{d}_{ob})$ and $\mathbf{N}(\mathbf{E}_{c',j}, 2\mathbf{d}_{ob})$ should always be an empty set.

Figure 15. Illustration of Intersection of $\mathbf{N}(\mathbf{q}_{ri}, 2\mathbf{d}_{ob})$ and $\mathbf{N}(\mathbf{E}_{c',j}, 2\mathbf{d}_{ob})$

Let $\mathbf{q}_n$ in [5] be a point bordered by the most explored area, and it is given as

$$\text{Let } \mathrm{q}_n = \arg \max_{q_{n*} \in B_{N2*}} \left(\mathrm{D}(q_{n*}) \cap \mathrm{E}_{c,\,\mathrm{i}}\right); \qquad (9)$$

if the number of $\mathbf{q}_n$ is more than one, then choose any of them as next point. Figure 16 demonstrates the geographical concept of a point bordered by the most covered spaces.



In this figure, since point 1 is bordered by three covered spaces while point 2 is bordered by only two covered spaces, the next move should be point 1 for $\mathbf{q}_{ri}$.

Figure 16. Illustration of Choosing Next Point for $\mathbf{q}_{ri}$

2. Grid–Based Implementation

The original algorithm from [5] is demonstrated in a generic two-dimensional representation, which is shown in Figure 17. As a step towards implementation, we reformulate the algorithm under two-dimensional uniformly gridded space. In other words, the direction that a robot can move under the original algorithm is defined initially as 360 degrees, while in our work the direction is restricted to four options; that is up, down, left, and right. The robot footprint with $0.5\mathbf{d}_r$ is illustrated in Figure 18.

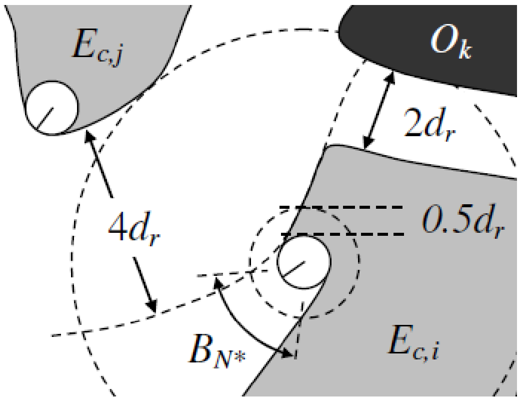
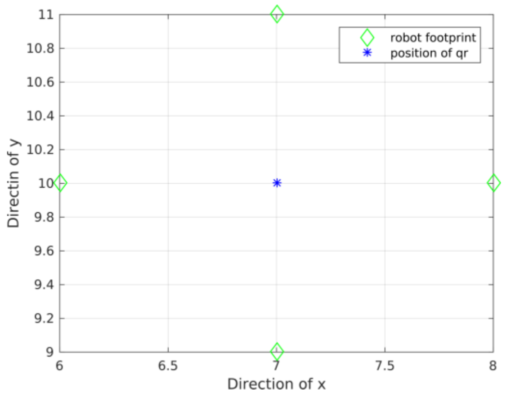


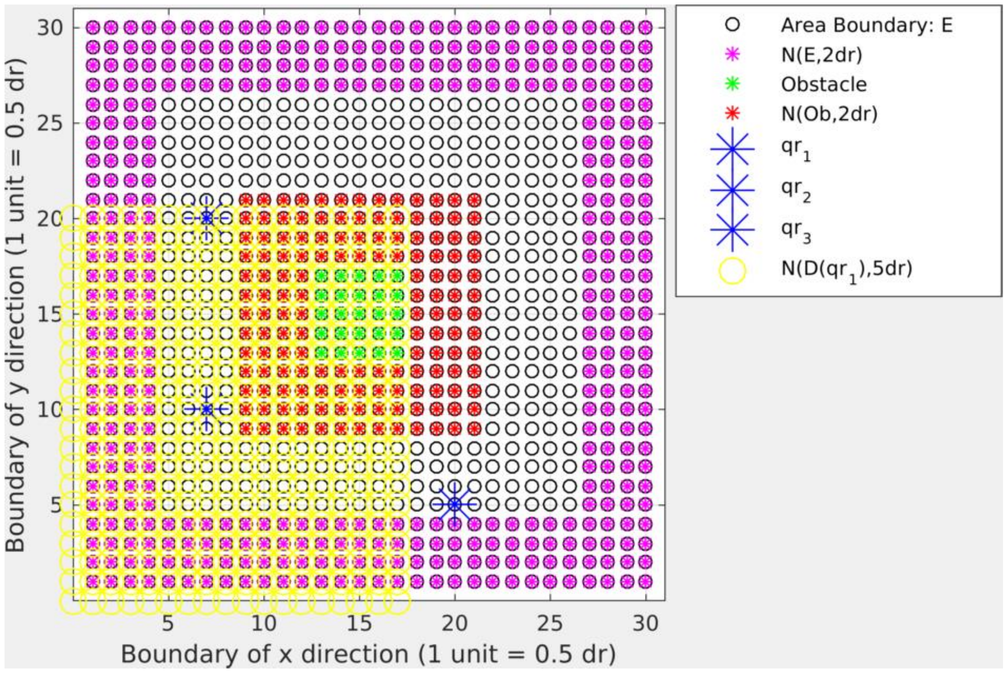
Figure 17. Direction that a Robot Can Turn in Original Algorithm. Source: [5].



The distance from blue star to green diamond equals $0.5\mathbf{d}_r$.

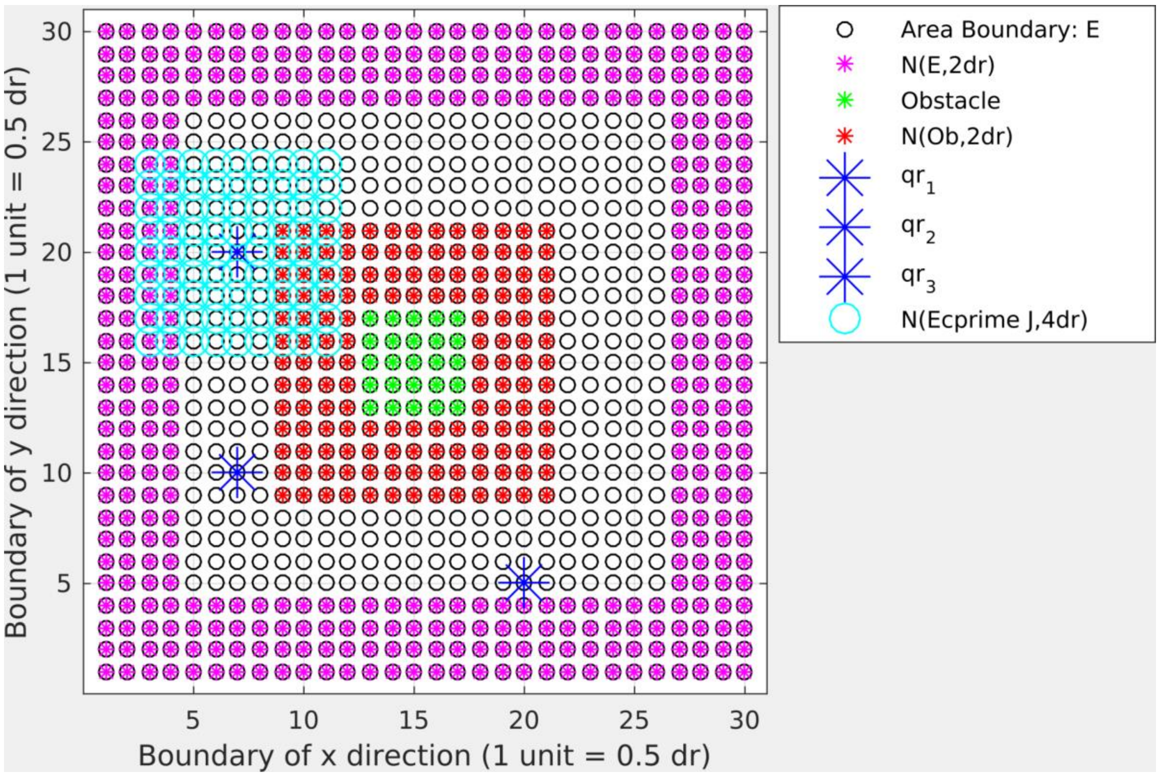
Figure 18. The Diameter of Robot Footprint Is $\mathbf{d}_r$

To visualize the possible and uncovered spaces to which the robot can go, Figure 19 and Figure 20 show the neighborhood of both the domain area and obstacle, and the neighborhood of $\mathbf{E}_{c',j}$ with a certain distance. The extension of the neighborhood centered at the robot serves as a detective fan, which is used to find whether there exists the position of another robot that intersects with the detective fan. If an intersection exists, then the position of the other robot is also the position of $\mathbf{E}_{c',j}$. Thus, the robot has to avoid getting close to the neighborhood established by the domain area, obstacle, and $\mathbf{E}_{c',j}$. In other words, the neighborhood occupied by others could be viewed as a forbidden zone for the robot. Figure 19 and Figure 20 display how to define the neighborhood of $\mathbf{E}_{c',j}$ step by step.



Robot #1, which is centered at [7, 10], looks for who is positioned at $\mathbf{E}_{c',j}$. And, because the position of robot #2 intersects with N(D(q_{r1}),5d_r) while robot#3 does not, the position of robot #2 is $\mathbf{E}_{c',j}$.

Figure 19. Possible and Uncovered Spaces for Robot #1 to Move (Part 1)

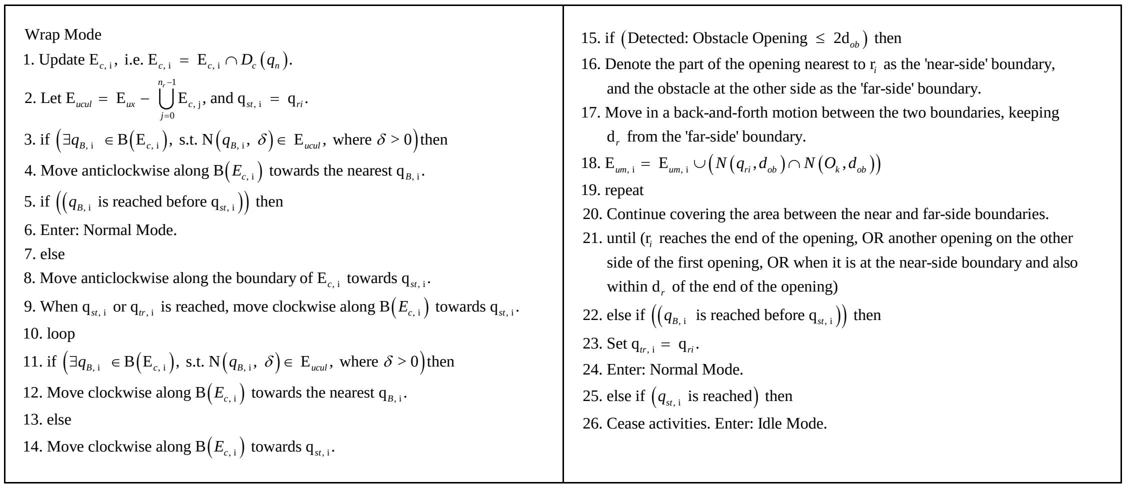


Once the $\mathbf{E}_{c',j}$ is defined, we can locate the blocked out area that robot #1 has to avoid. All in all, robot #1 cannot go to the neighborhood colored in magenta, red, and cyan.

Figure 20. Possible and Uncovered Spaces for Robot #1 to Move (Part 2)

C. WRAP MODE

If there is no available point for a robot to move under normal mode, the algorithm switches from normal mode to wrap mode. In wrap mode, $\mathbf{q}_{ri}$ follows the boundary established based on its explored points and goes counterclockwise. Once an unexplored point is found, wrap mode switches to normal mode until there is no unexplored point. The obstacle-opening problem is not discussed here, since when warships enter the sea channel they would avoid getting closer to obstacles that could lead to a collision or being stranded in shallow waters. Figure 21 shows the pseudo code of wrap mode.



```
Wrap Mode
1. Update $E_{c,i}$, i.e. $E_{c,i} = E_{c,i} \cap D_c(q_n)$.
2. Let $E_{ucul} = E_{ux} - \bigcup_{j=0}^{n_r-1} E_{c,j}$, and $q_{st,i} = q_{ri}$.
3. if $(\exists q_{B,i} \in B(E_{c,i})$, s.t. $N(q_{B,i}, \delta) \in E_{ucul}$, where $\delta > 0)$ then
4. Move anticlockwise along $B(E_{c,i})$ towards the nearest $q_{B,i}$.
5. if $((q_{B,i}$ is reached before $q_{st,i}))$ then
6. Enter: Normal Mode.
7. else
8. Move anticlockwise along the boundary of $E_{c,i}$ towards $q_{st,i}$.
9. When $q_{st,i}$ or $q_{tr,i}$ is reached, move clockwise along $B(E_{c,i})$ towards $q_{st,i}$.
10. loop
11. if $(\exists q_{B,i} \in B(E_{c,i})$, s.t. $N(q_{B,i}, \delta) \in E_{ucul}$, where $\delta > 0)$ then
12. Move clockwise along $B(E_{c,i})$ towards the nearest $q_{B,i}$.
13. else
14. Move clockwise along $B(E_{c,i})$ towards $q_{st,i}$.
15. if (Detected: Obstacle Opening $\leq 2d_{ob}$) then
16. Denote the part of the opening nearest to $r_i$ as the 'near-side' boundary,
    and the obstacle at the other side as the 'far-side' boundary.
17. Move in a back-and-forth motion between the two boundaries, keeping
    $d_r$ from the 'far-side' boundary.
18. $E_{um,i} = E_{um,i} \cup (N(q_{ri}, d_{ob}) \cap N(O_k, d_{ob}))$
19. repeat
20. Continue covering the area between the near and far-side boundaries.
21. until ($r_i$ reaches the end of the opening, OR another opening on the other
    side of the first opening, OR when it is at the near-side boundary and also
    within $d_r$ of the end of the opening)
22. else if $((q_{B,i}$ is reached before $q_{st,i}))$ then
23. Set $q_{tr,i} = q_{ri}$.
24. Enter: Normal Mode.
25. else if $(q_{st,i}$ is reached$)$ then
26. Cease activities. Enter: Idle Mode.
```

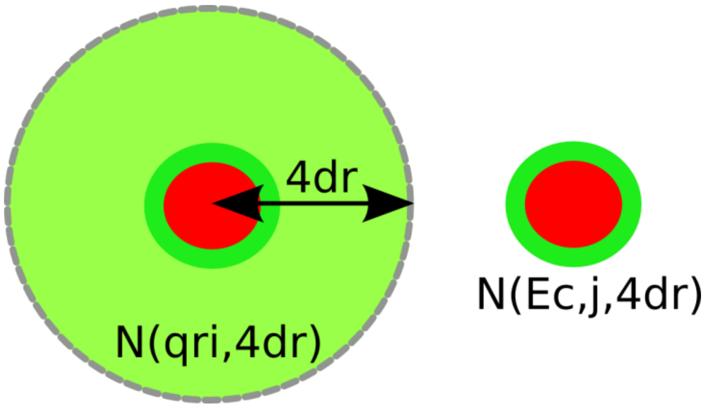
Figure 21. Pseudo Code of Wrap Code. Source: [5].

1. Description of the Basic Concept of Wrap Mode

Let $\mathbf{E}_{c,i}$ in [5] be redefined and expressed as

$$E_{c,i} = E_{c,i} \cap D_c(q_n), \tag{10}$$

where $\mathbf{E}_{c,i}$ is the explored area and $\mathbf{D}_c(\mathbf{q}_n)$ is a disc centered at $\mathbf{q}_{ri}$ with radius $0.5\mathbf{d}_r$. This will make $N(q_{ri}, 2d_{ob}) \cap N(E_{c,j}, 2d_{ob})$ different from the one we illustrated in Figure 15. With redefined $\mathbf{E}_{c,i}$, $N(q_{ri}, 2d_{ob}) \cap N(E_{c,j}, 2d_{ob})$ is displayed as Figure 22. The result will further affect the value of $\mathbf{E}_{um}$.



The result of $\mathbf{E}_{um}$ is different from the one in Equation 7 because the output of $N\left(\mathrm{E}_{c,j}, 2d_{ob}\right)$ is modified from Figure 21.

Figure 22. Illustration of Redefined Intersection of $\mathrm{N}(q_{ri}, 2d_{ob})$ and $N\left(\mathrm{E}_{c,j}, 2d_{ob}\right)$

Let $\mathbf{E}_{ucul}$ in [5] be the set after removing the areas explored by robots from the unexplored field and which is described as

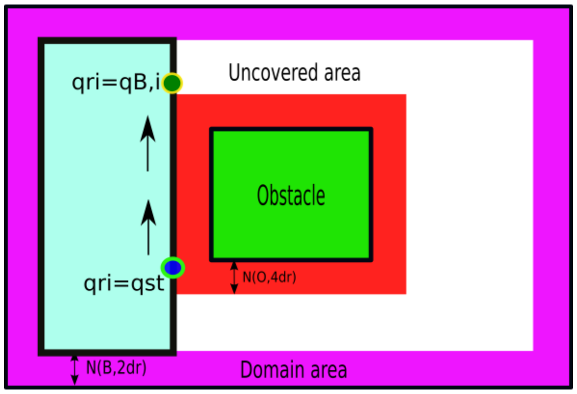
$$\mathrm{E}_{ucul} = \mathrm{E}_{ux} - \bigcup_{j=0}^{n_r - 1} E_{c,j}, \text{ and } \mathrm{q}_{st,i} = q_{ri} . \tag{11}$$

Furthermore, the position of $\mathbf{q}_{ri}$ represents a start point, $\mathbf{q}_{st}$, when the robot executes the algorithm under wrap mode.

Let $\mathbf{q}_{B,i}$ in [5] be a point on the boundary of the explored area, and the neighborhood of $\mathbf{q}_{B,i}$ exists on one or more than one unexplored points, which are a subset of $\mathbf{E}_{ucul}$. The expression of $\mathbf{q}_{B,i}$ is

$$\begin{array}{l} \text{if } \left(\exists q_{B,i} \in \mathrm{B}\left(\mathrm{E}_{c,i}\right), \text{ s.t. } \mathrm{N}\left(q_{B,i}, \delta\right) \in \mathrm{E}_{ucul}, \text{ where } \delta > 0\right) \\ \text{then move anticlockwise along the boundary of } \mathrm{E}_{c,i} \text{ towards the nearest } q_{B,i} \end{array} . \tag{12}$$

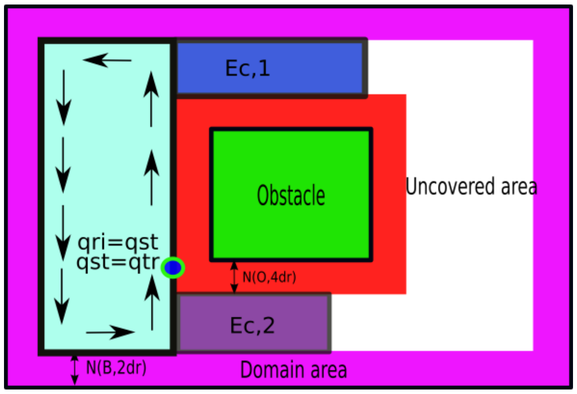
Once wrap mode is activated, the robot then moves counterclockwise looking for $\mathbf{q}_{B,i}$. If $\mathbf{q}_{B,i}$ is detected, the robot switches from wrap mode to normal mode. The graphical demonstration is shown in Figure 23.



The $\mathbf{q}_{ri}$ moves counterclockwise and toward the nearest $\mathbf{q}_{B,i}$, and then enters normal mode.

Figure 23. Illustration of $\mathbf{q}_{ri}$ Equals $\mathbf{q}_{st}$ under Wrap Mode and Moving toward Existing $\mathbf{q}_{B,i}$

If $\mathbf{q}_{B,i}$ cannot be located, then the robot moves along the boundary toward the $\mathbf{q}_{st}$. Once the $\mathbf{q}_{st}$ is reached, then the $\mathbf{q}_{st}$ serves as the transition point, $\mathbf{q}_{tr}$, which is the moment that the robot begins moving clockwise along the boundary to the $\mathbf{q}_{tr}$. The graphical demonstration is shown in Figure 24.



When entering wrap mode, $\mathbf{q}_{ri}$ moves counterclockwise looking for $\mathbf{q}_{B,i}$; if $\mathbf{q}_{B,i}$ cannot be found and as $\mathbf{q}_{ri}$ reaches $\mathbf{q}_{st}$ again, $\mathbf{q}_{st}$ would be redefined as $\mathbf{q}_{tr}$.

Figure 24. Illustration of $\mathbf{q}_{st}$ Becoming $\mathbf{q}_{tr}$ as $\mathbf{q}_{B,i}$ Cannot Be Located under Wrap Mode when $\mathbf{q}_{ri}$ Moves Counterclockwise

As the robot moves clockwise, it also searches for $\mathbf{q}_{B,i}$; if $\mathbf{q}_{B,i}$ is reached before $\mathbf{q}_{tr}$, then the robot switches from wrap mode to normal mode. If $\mathbf{q}_{B,i}$ cannot be located, then the robot moves along the boundary toward the $\mathbf{q}_{tr}$. Once $\mathbf{q}_{tr}$ is reached, the robot enters idle mode. The graphical demonstration is shown in Figure 25.

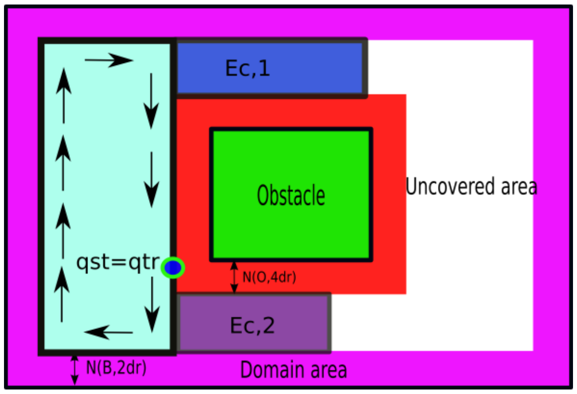


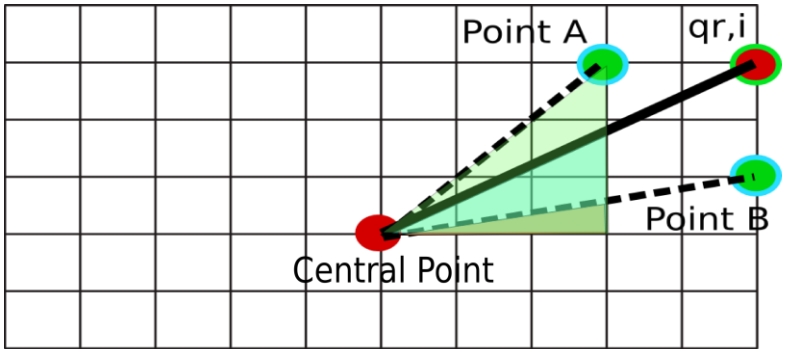
Figure 25. Illustration of $\mathbf{q}_{ri}$ Entering Idle Mode if $\mathbf{q}_{ri}$ Cannot Find $\mathbf{q}_{B,i}$ When Moving Clockwise from Transition Point and Reaching $\mathbf{q}_{tr}$ Again

D. DIRECTION OF MOVEMENT IN WRAP MODE ANALYSIS (COUNTERCLOCKWISE EXAMPLE)

In this thesis, the direction for robot to move for the next position is the most complicate challenge to overcome. The followings describe the primary concepts and methods to help us work out this problem.

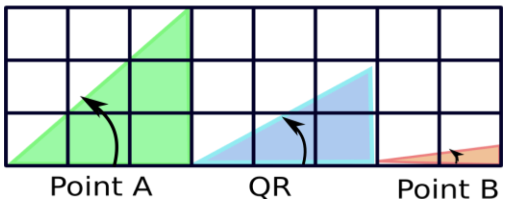
1. Basic Concept of Moving Counterclockwise

Generating direction in wrap mode requires distinguishing between counterclockwise and clockwise travel. Thus, three angles must be determined. The first angle is defined by the line between a central point and the current position of the robot, in which the central point is calculated by the area explored by the robot. There are two points located in front of and behind $\mathbf{q}_{ri}$, respectively, where $\mathbf{q}_{ri}$ stands for the current position of the robot. The second and third angles are defined by the lines between the central point and the points in front of and behind $\mathbf{q}_{ri}$. Signs (positive and negative) would be revealed after the second angle and third angle subtract to the first angle, respectively. The positive angle makes movement in wrap mode go clockwise while a negative angle makes it go counterclockwise. We use this feature to determine the direction of $\mathbf{q}_{ri}$. However, this characteristic is valid only when $\mathbf{q}_{ri}$ is on the boundary of its explored area. Figure 26 shows the way to determine the direction, and Figure 27 illustrates the comparison of angles.



Point A stands for the point in front of $\mathbf{q}_{ri}$ while point B is the point behind $\mathbf{q}_{ri}$.

Figure 26. Concept for $\mathbf{q}_{ri}$ to Determine a Direction



Angles of point A, $\mathbf{q}_{ri}$, and point B with respect to the mean value of explored points, respectively.

Figure 27. Comparison of Angles

2. Definition of "Required Points"

Because the value of an angle cannot guarantee $\mathbf{q}_{ri}$ to go counterclockwise or clockwise when $\mathbf{q}_{ri}$ does not have a connection with the boundary points of a domain area, we need to develop other approaches to make it work. That is the reason why "required points" arise. Required points mean the number of directions in which $\mathbf{q}_{ri}$

could choose to move. There are three different cases. We discuss the different cases and show how required points help us to make $\mathbf{q}_{ri}$ go correctly, as shown in Figure 28.

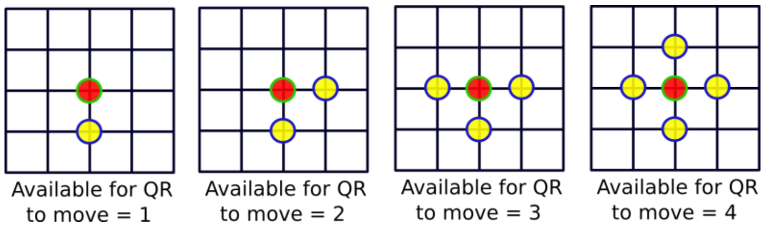
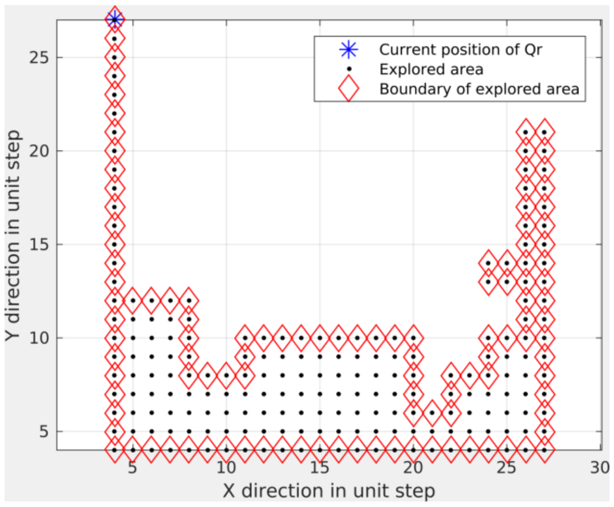

Available for QR to move = 1
Available for QR to move = 2
Available for QR to move = 3
Available for QR to move = 4

Figure 28. Four Cases for Directions in which $\mathbf{q}_{ri}$ Could Move

a. *Required Point Equals One Move*

When the required point is equal to one, the next point to which $\mathbf{q}_{ri}$ can move is only one. Just as the plot shows in Figure 29, the next point for $\mathbf{q}_{ri}$ will be the one under the current position of $\mathbf{q}_{ri}$.



Required point is equal to one as $\mathbf{q}_{ri}$ goes counterclockwise.

Figure 29. Required Point Is One Move

b. Required Points Equal Two Moves

In example 1 of Figure 30, the distance between point 1 and point 2 is equal to two moves, and there are three cases to discuss here. The first one occurs as point 1 and point 2 have the same y value. If the unexplored points are above $\mathbf{q}_{ri}$, then $\mathbf{q}_{ri}$ moves to the left. The second one happens when $\mathbf{q}_{ri}$ is next to the boundary of the domain area; in this situation, $\mathbf{q}_{ri}$ follows the "angle rule" to keep moving counterclockwise. The last one applies to the case when there is no boundary point around the sides. We could remove the previous point and have the other one continue going forward.

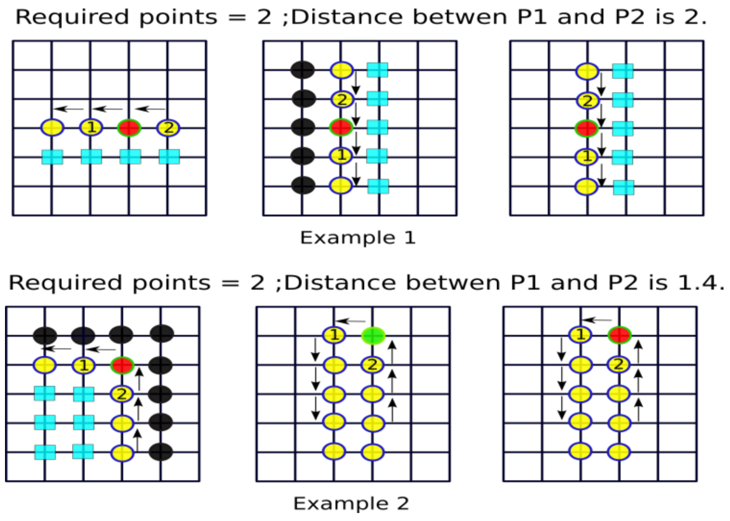
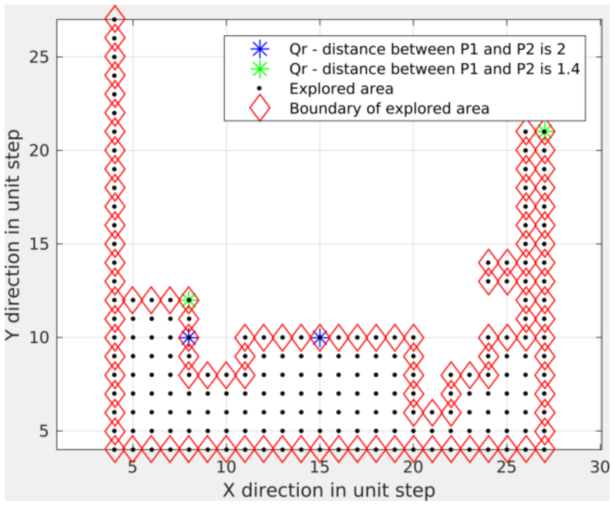

Required points = 2 ;Distance betwen P1 and P2 is 2.
Example 1
Required points = 2 ;Distance betwen P1 and P2 is 1.4.
Example 2

Figure 30. Two Different Distances as Required Points Equal Two Moves

In example 2 of Figure 30, the distance between point 1 and point 2 is about 1.4, and there are also three cases to discuss. In the first case, the $\mathbf{q}_{ri}$ is next to the boundary points of the domain area. Thus, $\mathbf{q}_{ri}$ could follow the angle rule to maintain the desired direction. Secondly, this is a much more complex situation because we have $\mathbf{q}_{B,i}$. We have to find out where the unexplored point is placed. If the unexplored point that has the same y value as $\mathbf{q}_{ri}$ is located on the right side of $\mathbf{q}_{ri}$, then $\mathbf{q}_{ri}$ should move left to keep moving counterclockwise, and vice versa. The last condition is easier to determine since removing the previous point is good enough to keep $\mathbf{q}_{ri}$ going forward. Figure 31 displays MATLAB representation of the case where the required points are equal to two moves and there are different distances between point 1 and point 2.



Required points equal two moves with different distances.

Figure 31. Required Points Equal Two Moves

c. *Required Points Equal Three Moves*

When the required points are equal to three moves, more complicated situations appear for us to consider. Thus, we need to come up with some ideas to help us analyze the situation thoroughly.

(1) Introduction of Different Neighborhood Patterns

There are three basic types of neighborhood patterns, and they are the "Big-square," "Small-square," and "Diamond-shape" patterns, which are shown in Figure 32.

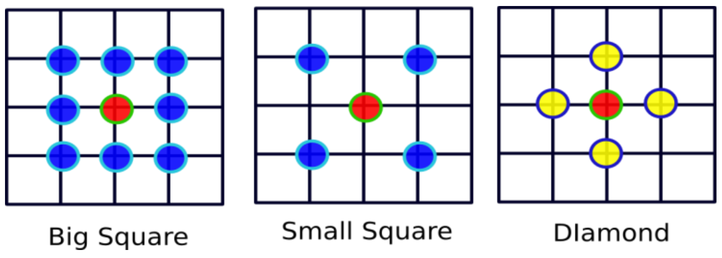


Figure 32. Neighborhood Patterns of $\mathbf{q}_{ri}$

(2) Four Conditions from Intersection of Small Square and Explored Points

The number of intersections of the small square (SS) and explored points (EP) leads to four possible conditions. These conditions are depicted in Figure 33, Figure 36, Figure 39, and Figure 42.

Zero point occupies the small square

Examples 1 to 4, discussed in the following paragraphs, are illustrated in Figure 34, and the specific condition in the MATLAB program is displayed in Figure 35.

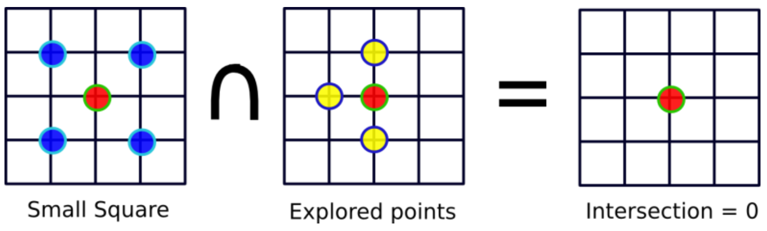


Figure 33. Intersection of SS and EP Equals Zero

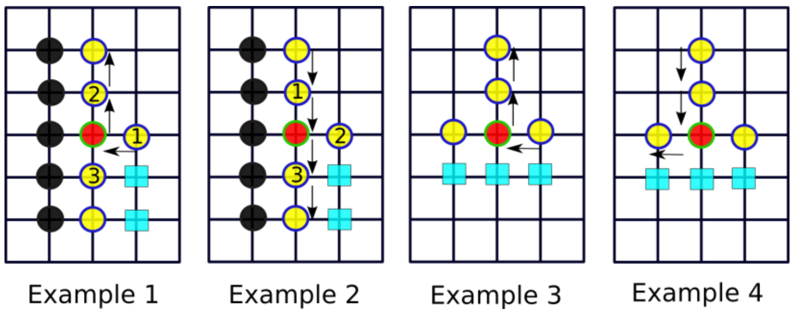


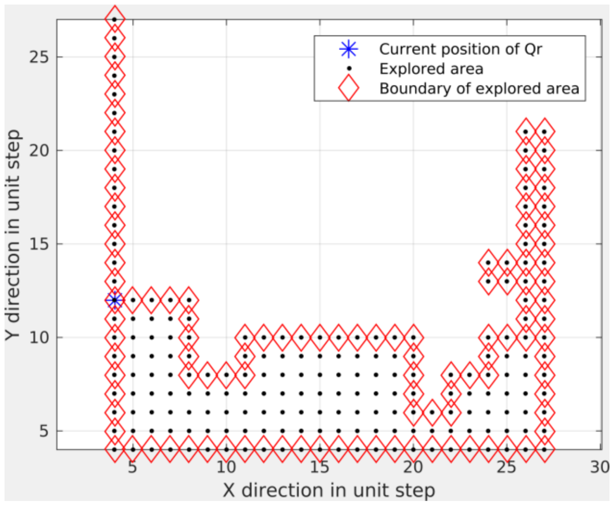
Figure 34. Required Points Equal Three Moves and Intersection with Small Square Equals Zero

Example 1 – The arrows show our desired route of $\mathbf{q}_{ri}$. This occurs as $\mathbf{q}_{ri}$ is next to the boundary of the domain area. Point 1 is the previous point. We do not want to go back; thus, we remove point 1. Since we have to explore each point on the boundary, we need to go up. The location of the unexplored point assists us to determine the next point for $\mathbf{q}_{ri}$. Point 3 has the explored point on its side; therefore, point 2 becomes our ideal next point.

Example 2 – After exploring the upper part of the points, we would like to go down now. We remove the previous point, which is point 1. Then, because $\mathbf{q}_{ri}$ is next to the boundary of the domain area, we could apply the angle rule to find the counterclockwise movement for this situation. Point 3 will be the next point.

Example 3 – In this example, there is no boundary of the domain area, but we still have to explore upward in this case. We remove the previous point and the point that has the same height as $\mathbf{q}_{ri}$ in order to determine the next position.

Example 4 – After exploring upward points, we need to find the next one. If the height of the points explored by other robots is above the height of $\mathbf{q}_{ri}$, the robot has to go left to guarantee counterclockwise direction and vice versa.



Required points equal Three Moves as $\mathbf{q}_{ri}$ goes Counterclockwise.

Figure 35. Zero Point Occupies Small Square under the Required Points Equal Three Moves Condition

One point occupies at small square

The number of intersections of the small square (SS) and explored points (EP) leads to four possible conditions. The second condition is depicted in Figure 36.

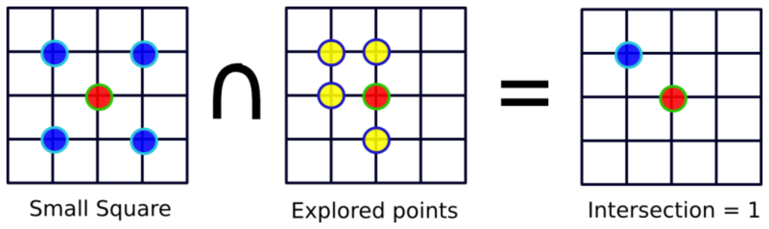


Figure 36. Intersection of SS and EP Equals One Move

The black points shown in Figure 37 represent the boundary points in the domain area. In this situation, we remove the point that has the same height as $\mathbf{q}_{ri}$ (point 1) and have the other two points follow the angle rule. If we want the $\mathbf{q}_{ri}$ to move counterclockwise, point 2 will be chosen as the next position of $\mathbf{q}_{ri}$, and vice versa. Figure 38 demonstrates the specific condition using the MATLAB program.

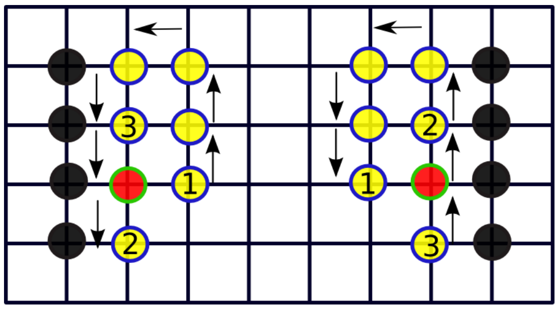
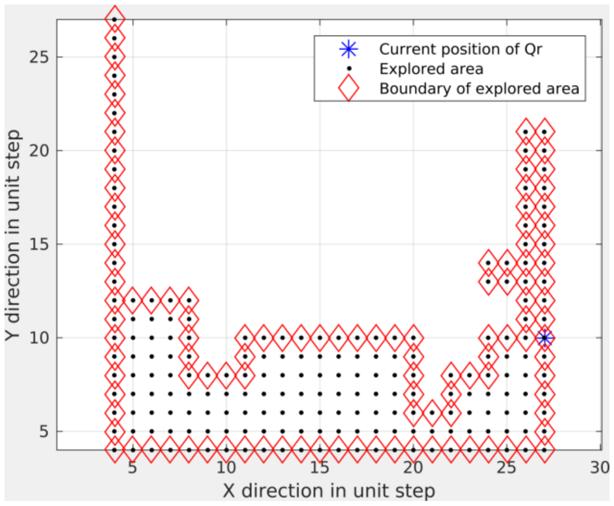


Figure 37. Required Points Equal to Three Moves and Intersection with Small Square Is One Move



Required points equal three moves as $\mathbf{q}_{ri}$ goes counterclockwise.

Figure 38. One Point Occupies the Small Square under the Required Points Equal Three Moves Condition

Two points occupy a small square

As stated previously, the number of intersections of the small square (SS) and explored points (EP) leads to four possible conditions. The third condition is depicted in Figure 39. Figure 40 demonstrates how does robot move to next position under this situation.

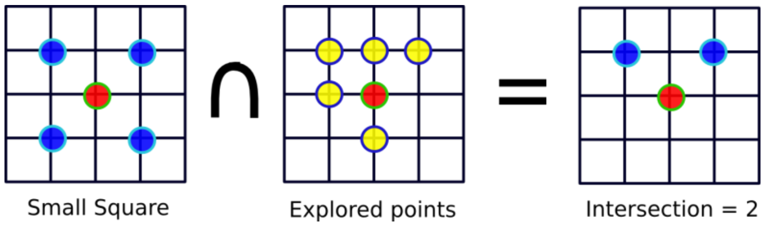


Figure 39. Intersection of SS and EP Equals Two Moves

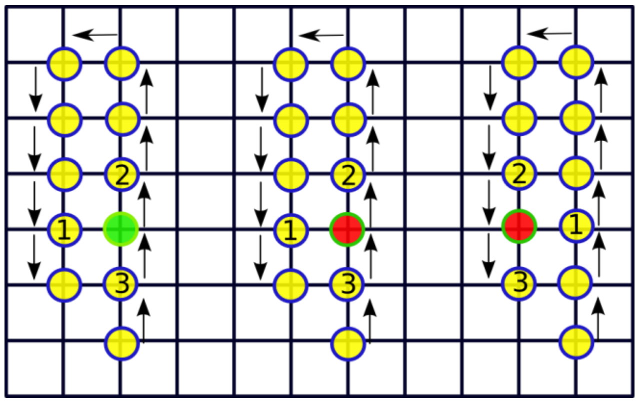
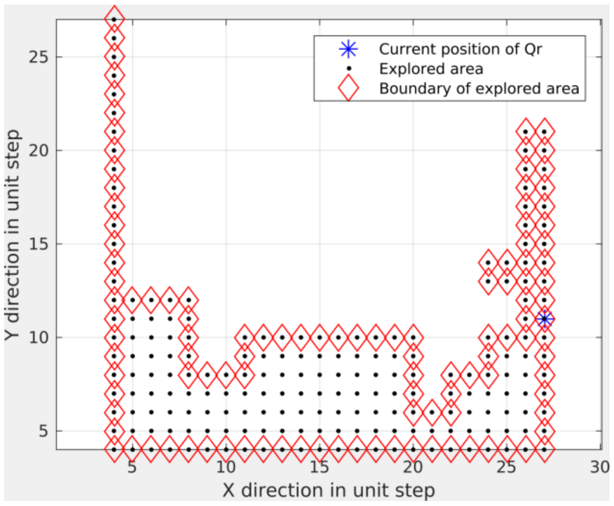


Figure 40. Required Points Equal Three Moves and Intersection with Small Square Equals Two Moves

The $\mathbf{q}_{B,i}$ is the unexplored point before being discovered by $\mathbf{q}_{ri}$ when it is under wrap mode. If the current position of $\mathbf{q}_{ri}$ was the unexplored point previously, $\mathbf{q}_{ri}$ has to determine the correct direction to go. Under this condition, first, we remove the point that has the same altitude as $\mathbf{q}_{ri}$. Then we let $\mathbf{q}_{ri}$ go up if there is no explored point on its right and go down when there is no explored point on its left. Those features are illustrated in Figure 40, and the specific condition simulated in the MATLAB program is shown in Figure 41.



Required point equals three moves as $\mathbf{q}_{ri}$ goes counterclockwise.

Figure 41. Two Points Occupy the Small Square under the Required Points Are Three Moves Condition

Three points occupy a small-square

. The last condition is illustrated in Figure 42. Two examples of another condition of the number of intersections of the small square (SS) and explored points (EP) are shown in Figure 43, and the specific condition simulated in the MATLAB program is shown in Figure 44.

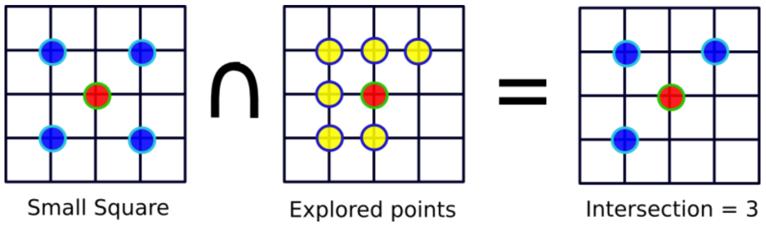


Figure 42. Intersection of SS and EP Equals Three Moves

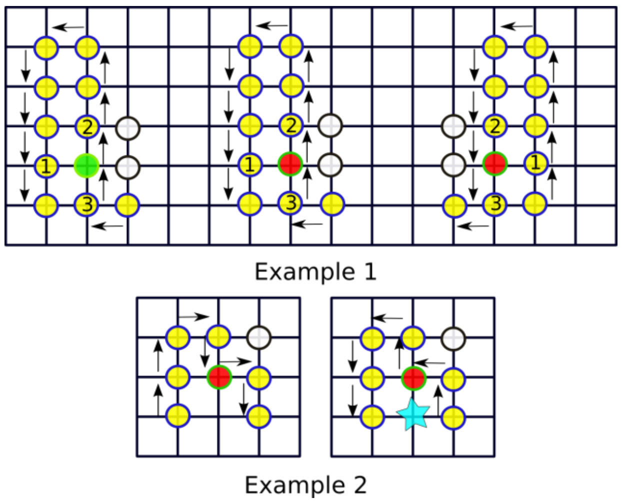
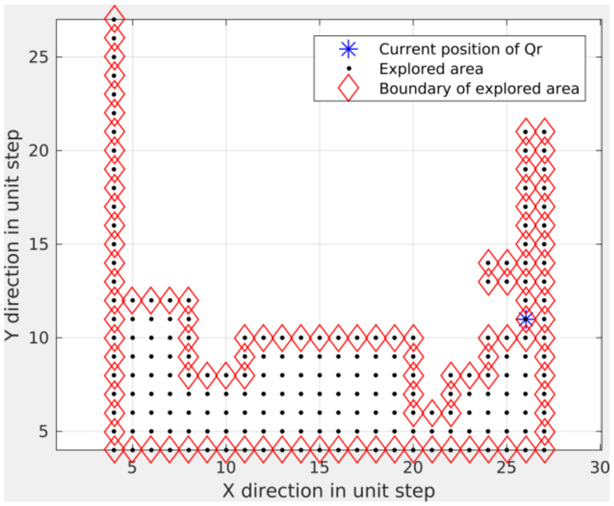


Figure 43. Required Points Equal Three Moves and the Intersection with Small Square Equals Three Moves

Example 1 – If we have $\mathbf{q}_{B,\mathrm{i}}$ under this situation, we remove the point that has the same height as $\mathbf{q}_{ri}$ first, which is point 1. Then we find out which side of the unexplored points are on; if they are on the right side, $\mathbf{q}_{ri}$ goes up. By contrast, $\mathbf{q}_{ri}$ goes down when they are on the left side. If we do not have $\mathbf{q}_{B,\mathrm{i}}$, we could just remove the previous point and the point that has the same height as $\mathbf{q}_{ri}$ to keep $\mathbf{q}_{ri}$ going counterclockwise.

Example 2 – In example 2, the figure on the left is easier because we can take out the previous point and the third point count from the end to have the desired point remain. The figure on the right, however, depicts a much more complex case. We do the same to delete the previous point. Then we have to find out the location of the star point. If the star point has the same x value as $\mathbf{q}_{ri}$, then we pick the point that has the same x value as $\mathbf{q}_{ri}$, too.



Required point equals three moves as $\mathbf{q}_{ri}$ goes counterclockwise.

Figure 44. Three Points Occupy Small Square under the Required Points Equal Three Moves Condition

d. Required Points Equal Four Moves

Figure 45 displays this condition graphically, including the position of $\mathbf{q}_{ri}$, the desired direction and the distribution of other points. We apply a big square neighborhood pattern as our base and remove the explored points around the $\mathbf{q}_{ri}$. Usually we will have one point on the left, which becomes the critical point. The procedure to locate the critical point is shown in Figure 46.

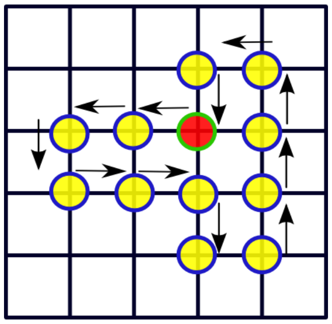
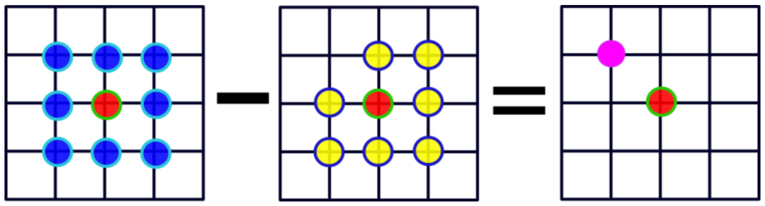


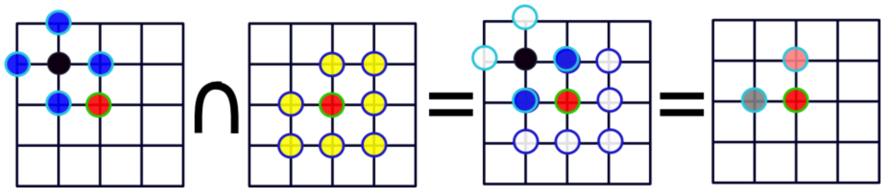
Figure 45. Required Points Equal Four Moves



Find critical point by removing explored points on the boundary from the big square neighborhood pattern.

Figure 46. Demonstration of Finding the Critical Point

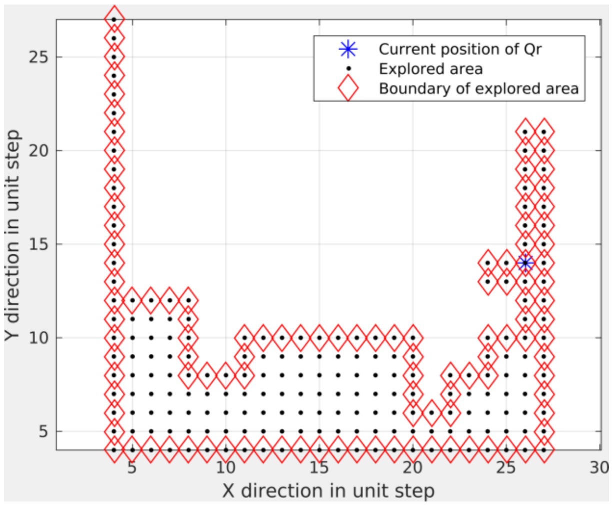
A diamond-shaped neighborhood pattern around the critical point would be used and then we would look for its intersection points with the explored points around the $\mathbf{q}_{ri}$.
The procedure to find the next possible points is shown in Figure 47.



Find possible moves by locating the intersection of the diamond-shaped neighborhood centered at the critical point and the explored area on the boundary centered at $\mathbf{q}_{ri}$.

Figure 47. Demonstration of Locating Next Moves

We will have two points after the previous step. One is the previous position of $\mathbf{q}_{ri}$ and the other is the desired next position for $\mathbf{q}_{ri}$. What we have to do is to remove the previous point and have the other point remain. The specific condition simulated in the MATLAB program is illustrated in Figure 48.

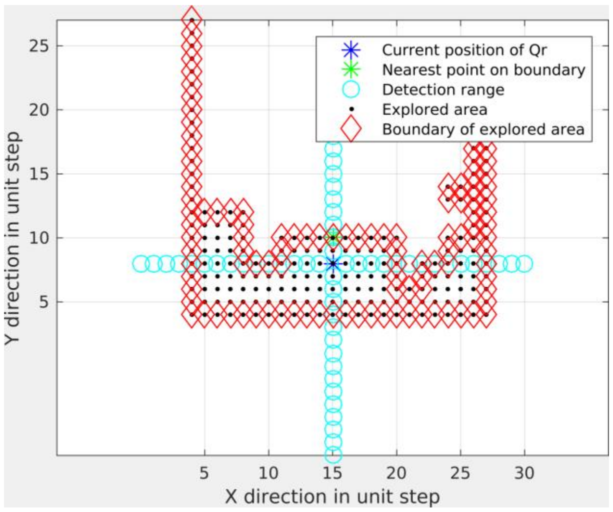


Required point equals four moves as $\mathbf{q}_{ri}$ goes counterclockwise.

Figure 48. Required Points Equal Four Moves

e. Required Point Equals Zero

When executing the aforementioned rules, the required point equals zero when there are no more boundary points around $\mathbf{q}_{ri}$. Some of the boundary points disappear as the unexplored points surrounded by them have been covered. As a result, $\mathbf{q}_{ri}$ has to find a way back to the nearest boundary to carry out the algorithm. Since $\mathbf{q}_{ri}$ can only go up, down, left, and right, we apply and extend the length of the diamond-shaped neighborhood to find the nearest point on the boundary of its explored area, and drive the $\mathbf{q}_{ri}$ to move to the nearest point. The specific condition simulated in the MATLAB program is shown in Figure 49.



Required point equals zero condition.

Figure 49. Required Point Equals Zero

3. Concept to Go Clockwise

The logical rules to execute the clockwise direction are the opposite of the rules to perform the counterclockwise direction.

E. TWO-DIMENSIONAL GRID-BASED IMPLEMENTATION

In this section, we display how the normal mode and combination of normal mode and wrap mode do in MATLAB program, respectively.

1. Normal Mode

Figure 50 shows the implementation of normal mode when exploring the area.

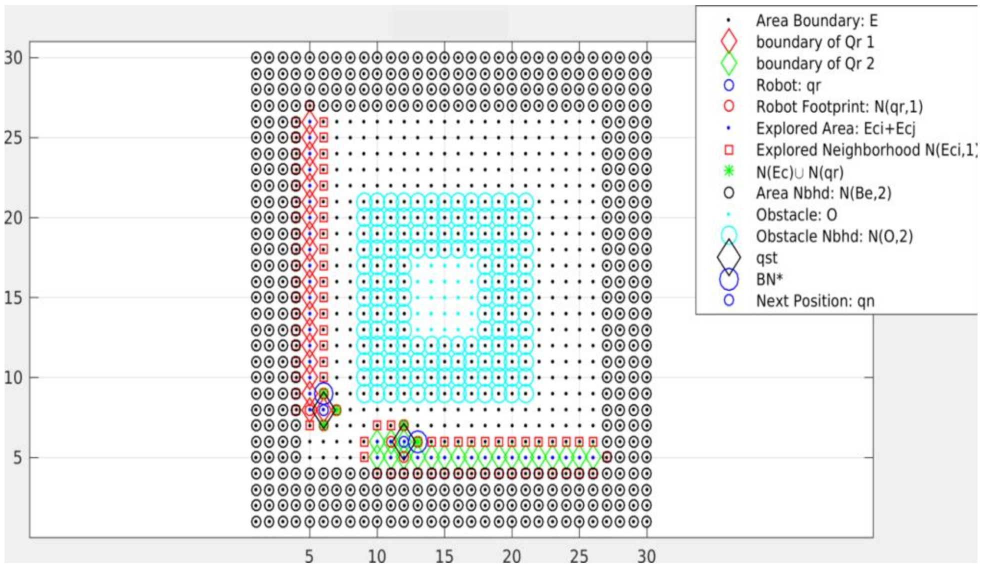


Figure 50. Normal Mode Performed by Two Robots

2. Normal Mode + Wrap Mode

Figure 51 shows the implementation of normal mode when exploring the area.

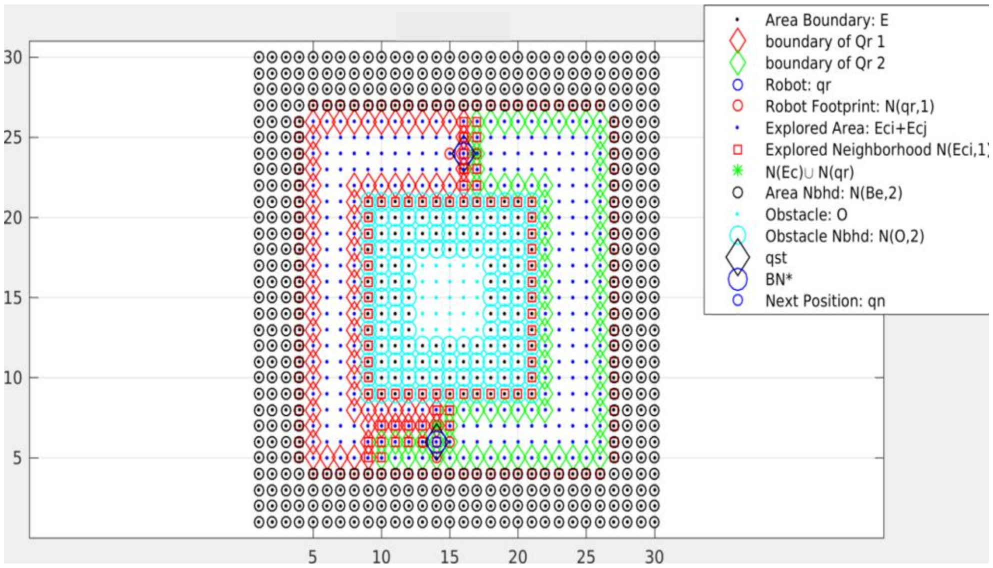


Figure 51. Robots Switch between Normal Mode and Wrap Mode to Complete Coverage

F. DEVELOPED MATLAB CODE FOR THE ALGORITHM

All the developed MATLAB codes are saved on the following website at gitlab. https://gitlab.nps.edu/bsbingha/multirobot_coverage/tree/master/matlab.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SIMULATION EXPERIMENT

A. DESCRIPTION OF EXPERIMENTS

In this work, there are nine experiments, shown in Table 8. The number of obstacles and the number of robots are independent variables. Both of them vary from one to three. We perform the experiment using a different number of obstacles and robots and evaluate the number of time steps as a measure of system performance. The following lists the characteristics of these experiments: Let N denotes the number of robots and O denote the number of obstacles, which are expressed as $\mathbf{Exp}(N,O)$ in Table 7.

Table 7. Experiment Matrix

		Number of obstacle		
		1	2	3
number of robot	1	Exp(1,1)	Exp(1,2)	Exp(1,3)
	2	Exp(2,1)	Exp(2,2)	Exp(2,3)
	3	Exp(3,1)	Exp(3,2)	Exp(3,3)

1. Domain Area

The domain area is a 30 by 30 grid. Based on the algorithm, each point stands for $0.5\mathbf{d}_r$ and every robot has to keep $2\mathbf{d}_r$ away from the domain boundary. Figure 52 presents the unexplored points as well as the neighborhood field within the domain area.

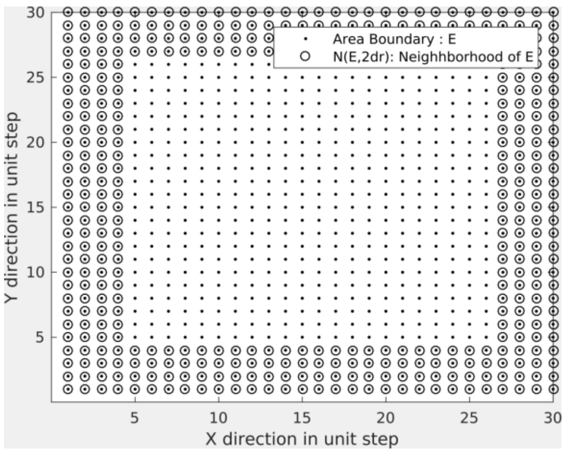


Figure 52. Unexplored Points and Neighborhood Field of Domain Area

2. Obstacles

Robots also have to keep $2\mathbf{d}_r$ from obstacles. Thus, we construct a neighborhood with a thickness of four points around each obstacle. The outline of the obstacle is depicted by points. Even though we have different cases of varying numbers of obstacles, we maintain the same total area for the obstacles for all experiments. Subsequently, we arrange one obstacle with area of sixteen, two obstacles with area of eight, respectively, and three obstacles are composed of one area of eight and two areas of four. Different numbers of obstacles cases are shown in Figure 53, Figure 54, and Figure 55.

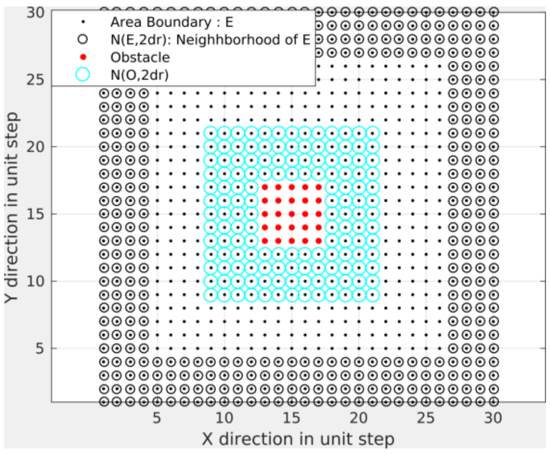


Figure 53. One-Obstacle Environment

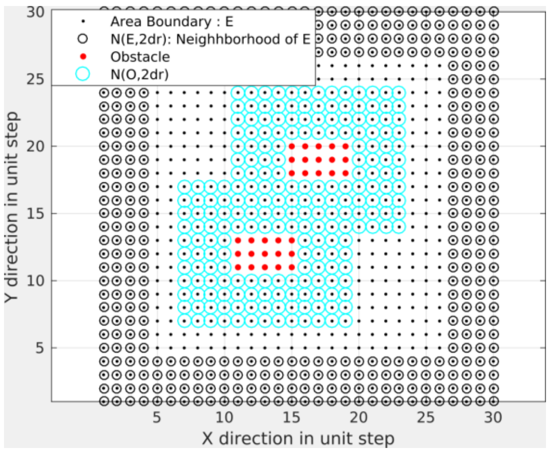


Figure 54. Two-Obstacle Environment

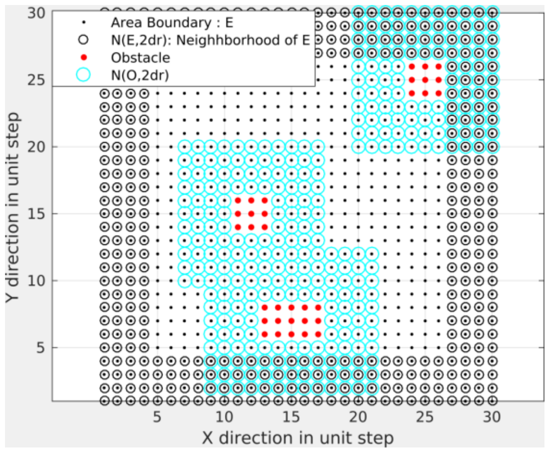


Figure 55. Three-Obstacle Environment

Keeping the area the same does not guarantee that the value of the unexplored points will be equal. In other words, the area an obstacle has does not have an essential influence on the coverage algorithm in conducting the MCM mission, but the number of obstacles will since the unexplored points decrease significantly due to neighborhood points.

3. Number of Robots

The number of robots varies from one to three. The first robot is located at $[5, 26]$, the second robot positioned at $[26, 5]$, and the third one started from $[19, 26]$. The same initial robot locations are used for all trials.

4. Experiment Scenarios

The prior knowledge of environments is unknown to all robots. Thus, robots will not have information about the location of uncovered spaces. However, the data about explored points visited by all robots is shared and used to implement the algorithm. The exact cellular decomposition is applied to the scenario. Robots move back-and-forth to visit points under normal mode and move along the boundary of the explored area under wrap mode. In wrap mode, the robot goes counterclockwise from $\mathbf{q}_{st}$ to $\mathbf{q}_{st}$ and moves clockwise from $\mathbf{q}_{tr}$ to $\mathbf{q}_{tr}$, as it looks for ignored unexplored points. Subsequently, $\mathbf{q}_{st}$ becomes $\mathbf{q}_{tr}$ after robot meets $\mathbf{q}_{st}$ again while moving counterclockwise. The algorithm is terminated once the robot arrives at $\mathbf{q}_{tr}$ again when going clockwise.

5. Evaluation Methods

There are two critical performance metrics that we desire to observe. One is the number of time steps robots take in each environment to implement the algorithm. This is directly proportional to the completion time steps for the mission. The other is the ratio of area covered to total area, where complete coverage is indicated by a ratio of 1.0.

B. PREDICTION

We listed the total number of free grid cells in each experiment as shown in Table 8. If a robot knows the environment in advance, it should take 315 time steps to finish the coverage. With the number of robots being more than one, we just divide the free grid cells by the number of robots. We set the time steps as shown in Table 9 as our criteria and the ratio is 1.0 under this condition.

Table 8. Total Number of Free Grid Cells

		Number of obstacles		
		1	2	3
number of robots	1	315	234	126
	2	315	234	126
	3	315	234	126

Table 9. Lower Bound of Time Steps for Each Experiment (Min # Steps)

		Number of obstacles		
		1	2	3
number of robots	1	315	234	126
	2	158	117	63
	3	105	78	42

C. EXPERIMENTS

Here, we reveal our result in each experiment. The result is displayed in MATLAB plot with the last step of each calculation.

1. One Robot with One Obstacle $(1,1)$

This is a single robot coverage implementation. As shown in Figure 56, the robot took 435 steps to complete the coverage and that coverage was 100% complete.

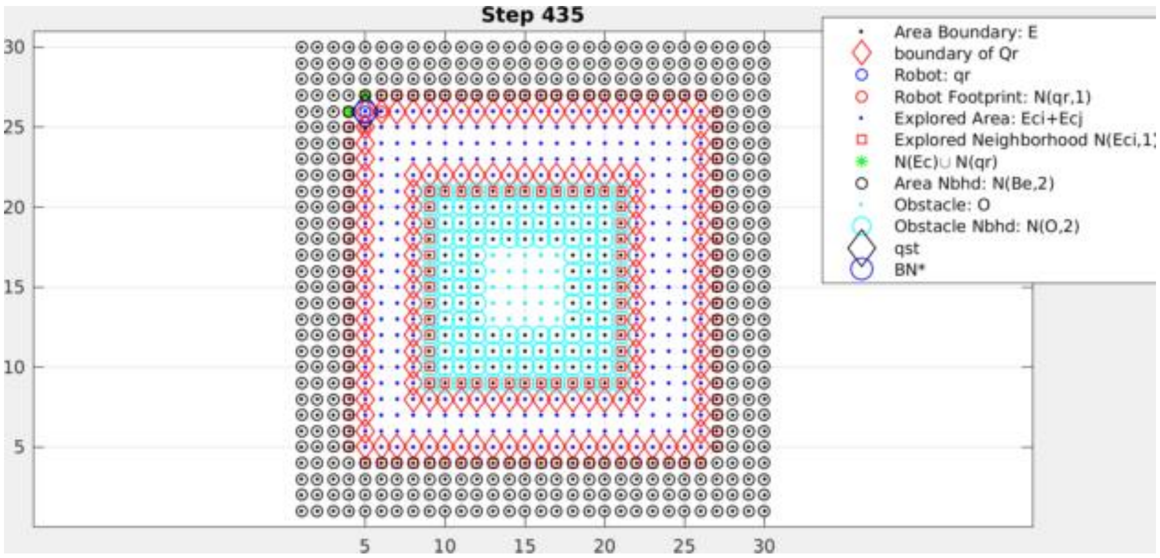


Figure 56. Coverage Algorithm Implemented by One Robot in One-Obstacle Environment

2. One Robot with Two Obstacles $(1,2)$

This is a single robot coverage implementation. As shown in Figure 57, the robot took 315 steps to complete the coverage. Increasing the number of neighborhood points is reflected in the reduction in steps.

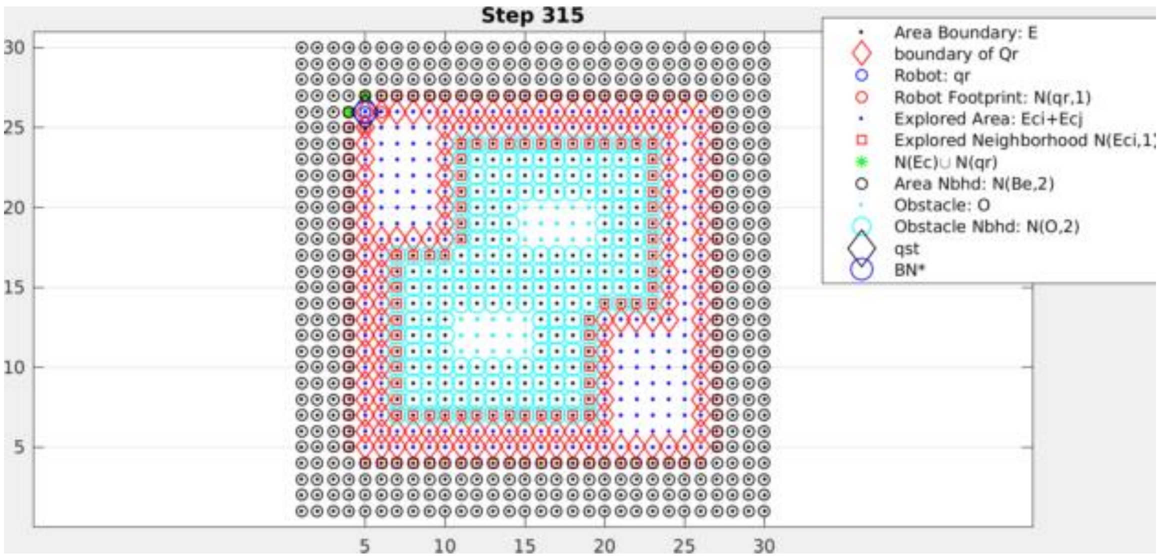


Figure 57. Coverage Algorithm Implemented by One Robot in Two-Obstacle Environment

3. One Robot with Three Obstacles $(1,3)$

This is a single robot coverage implementation. As shown in Figure 58, the robot took 335 steps to complete the coverage. Increasing the number of neighborhood points leads to some loss of steps, but the extended boundary increases the number of steps catastrophically. The obstacles connected to the boundary of the domain area boost the number of boundary points of the area explored by the robot. When it implements the algorithm under wrap mode, the robot has to go along the boundary of its own explored area both counterclockwise and clockwise. In comparison to experiment 1 and experiment 2, whose obstacles are not connected to the boundary of the domain area, experiment 3 has a longer boundary of explored area due to this connection.

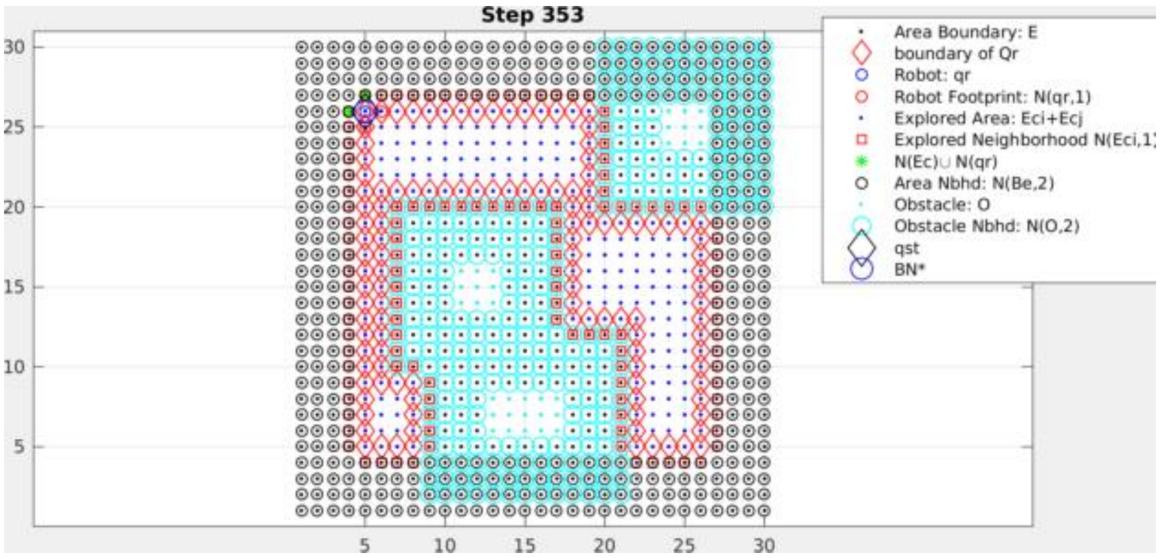


Figure 58. Coverage Algorithm Implemented by One Robot in a Three-Obstacle Environment

4. Two Robots with One Obstacle $(2,1)$

This is a two robot coverage implementation. As shown in Figure 59, the robot takes 334 steps to complete the coverage. As predicted, the number of steps declined because of the increase in the number of robots. And, when the number of robots reaches two, the behavior of colleague avoidance could be observed, which is shown in Figure 60. Ignored unexplored area occurs under the colleague avoidance condition. It will be visited if other robots are far enough away from it and after the robot completes a circle by moving counterclockwise along the boundary of its own explored area. The scenario is illustrated in Figure 61. This feature results in an increase in steps. From Figure 60 and Figure 61, we observe a difference of almost 100 steps.

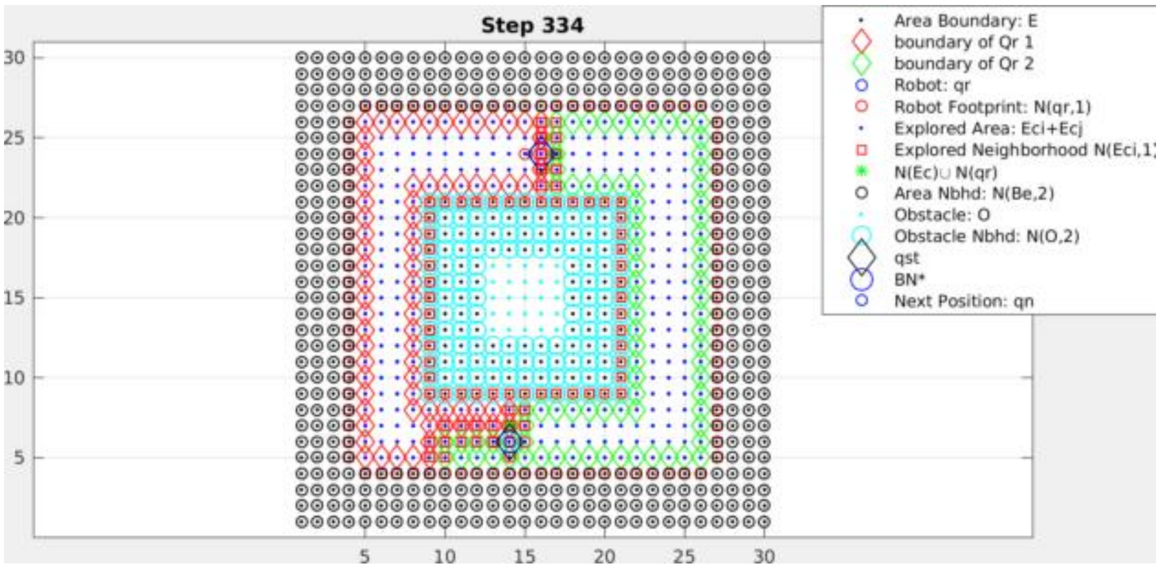


Figure 59. Coverage Algorithm Implemented by Two Robots in a One-Obstacle Environment

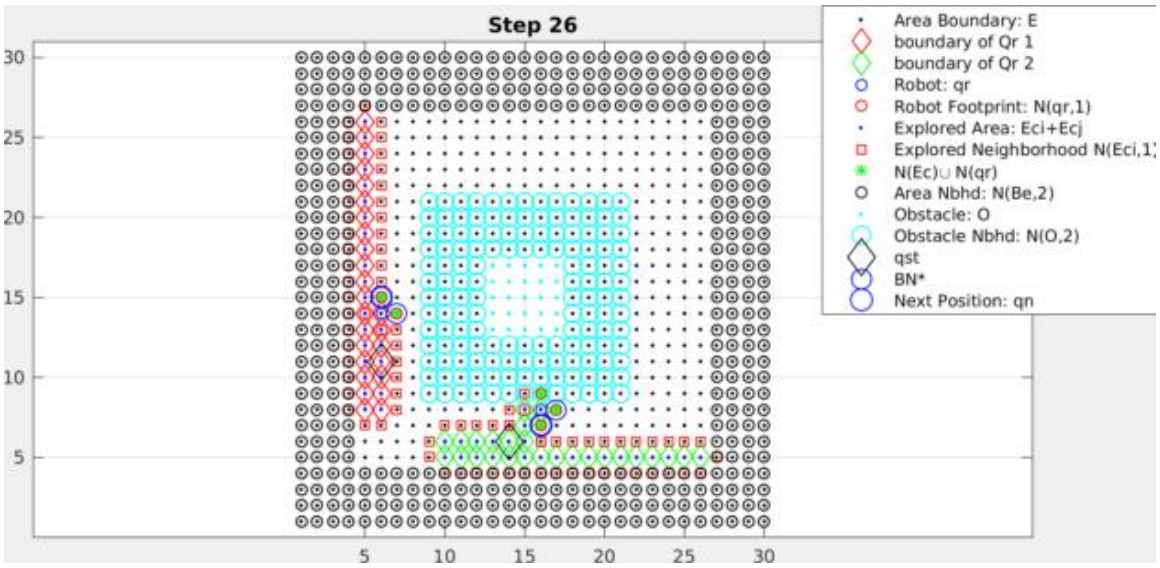


Figure 60. Colleague Avoidance Behavior

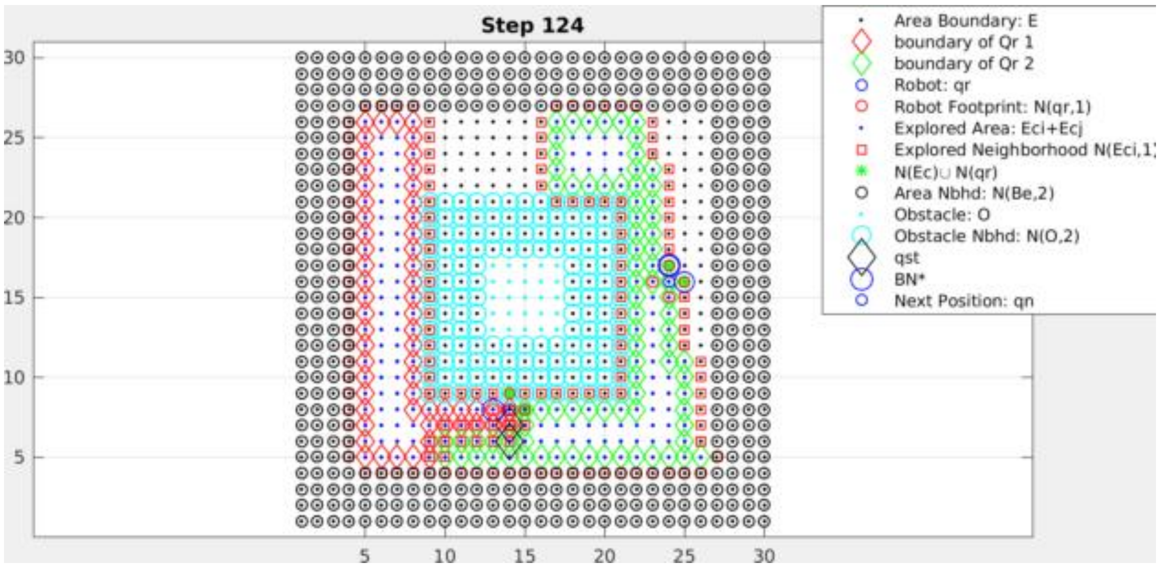


Figure 61. Ignored Unexplored Points Visited by One Robot as the Other Robot Is Far Away

5. Two Robots with Two Obstacles $(2,2)$

This is a two robot coverage implementation. As shown in Figure 62, it takes 328 steps to complete the coverage.

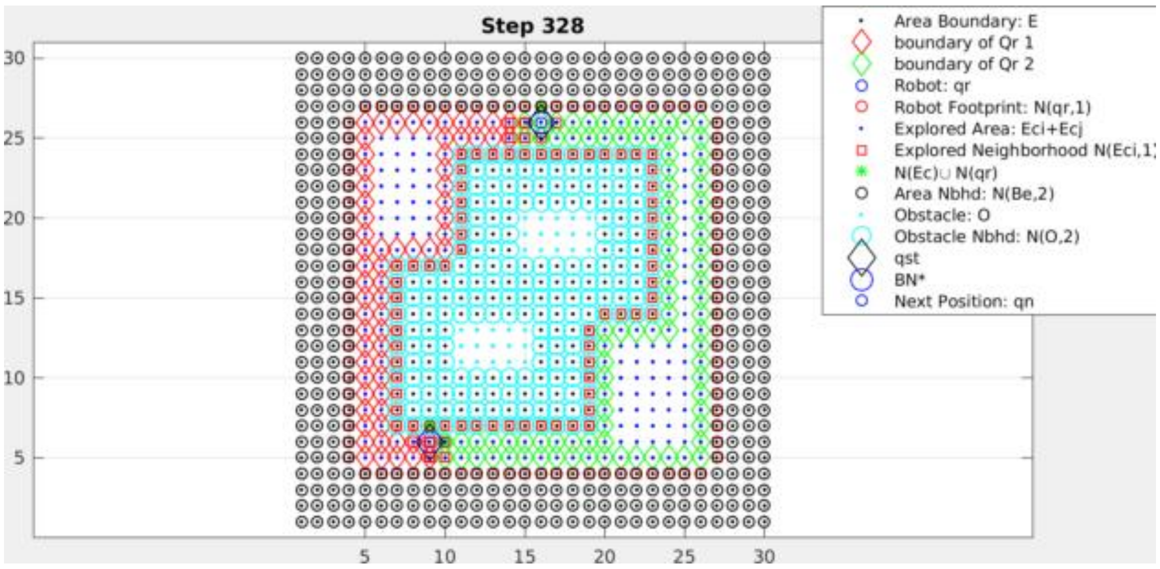


Figure 62. Coverage Algorithm Implemented by Two Robots in a Two-Obstacle Environment

6. Two Robots with Three Obstacles $(2,3)$

This is a two robot coverage implementation. As shown in Figure 63, it takes 259 steps to complete the coverage. Compared with experiment 4 and experiment 5 that have two areas in which robots interfere with each other due to the behavior of colleague avoidance, experiment 6 only has one interference area. Also from the comparison, we find it is the number of interference areas that affects the efficiency of the coverage algorithm.

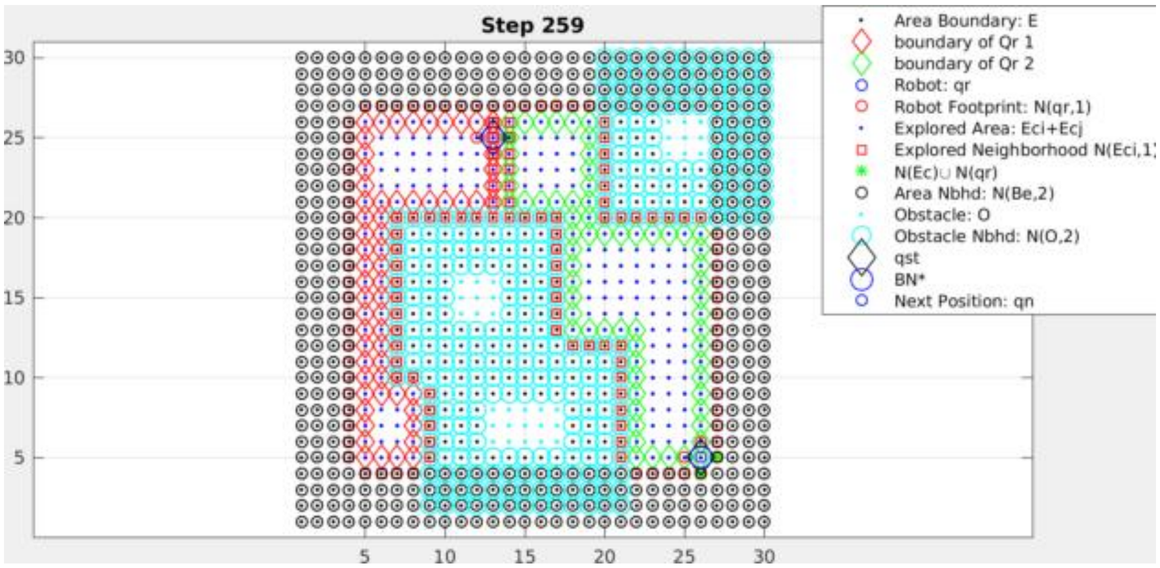


Figure 63. Coverage Algorithm Implemented by Two Robots in a Three-Obstacle Environment

7. Three Robots with One Obstacle $(3,1)$

This is a three robot coverage implementation. As shown in Figure 64, it took 203 steps to complete the algorithm. Yet, the coverage could not be completed for two reasons. The first factor was the colleague avoidance behavior, which is fully explained in the experiment. The second one was the rule of algorithm. The robot goes counterclockwise and searches for ignored unexplored points along the boundary of the explored area for a circle. If another robot is close enough that it has to avoid the robot again, then this ignored area will be overlooked again for the same reason. Incomplete coverage happens when one robot is still too close to another while patrolling clockwise along the boundary of the explored area.

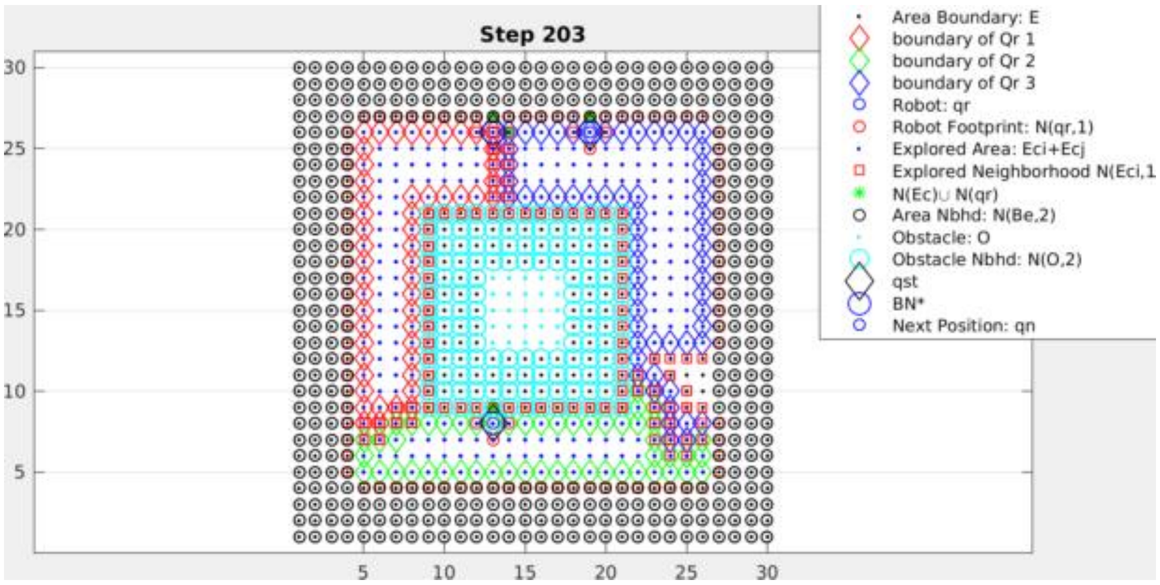


Figure 64. Coverage Algorithm Implemented by Three Robots in a One-Obstacle Environment

8. Three Robots with Two Obstacles $(3,2)$

This is a three robot coverage implementation. As shown in Figure 65, it takes 233 steps to complete the coverage.

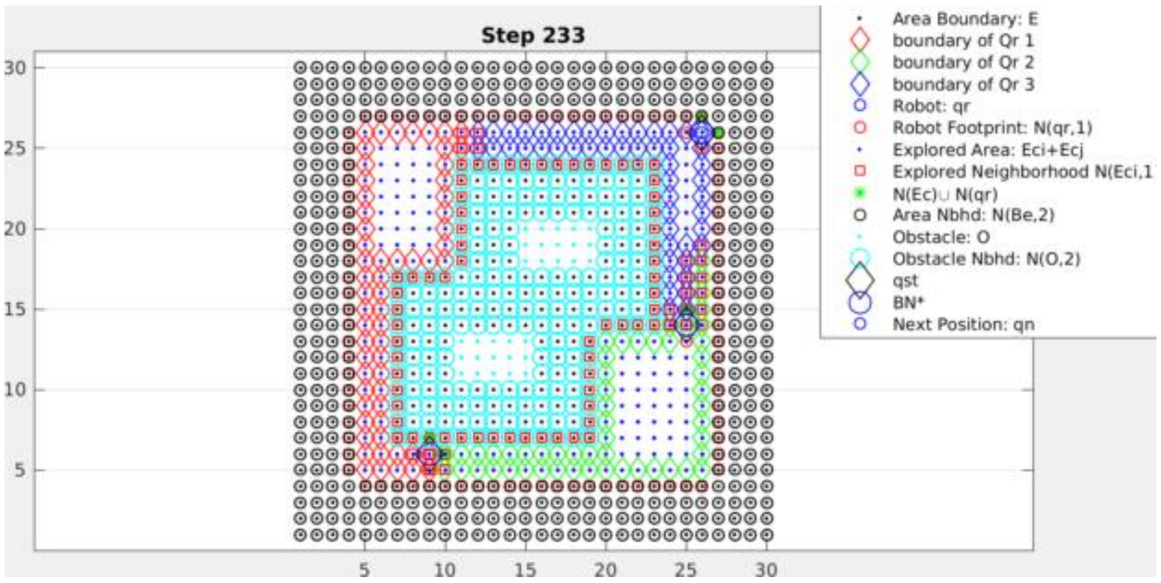


Figure 65. Coverage Algorithm Implemented by Three Robots in a One-Obstacle Environment

9. Three Robots with Three Obstacles $(3,3)$

This is a three robot coverage implementation. As shown in Figure 66, it took 316 steps to complete the algorithm. The coverage could not be completed in this experiment.

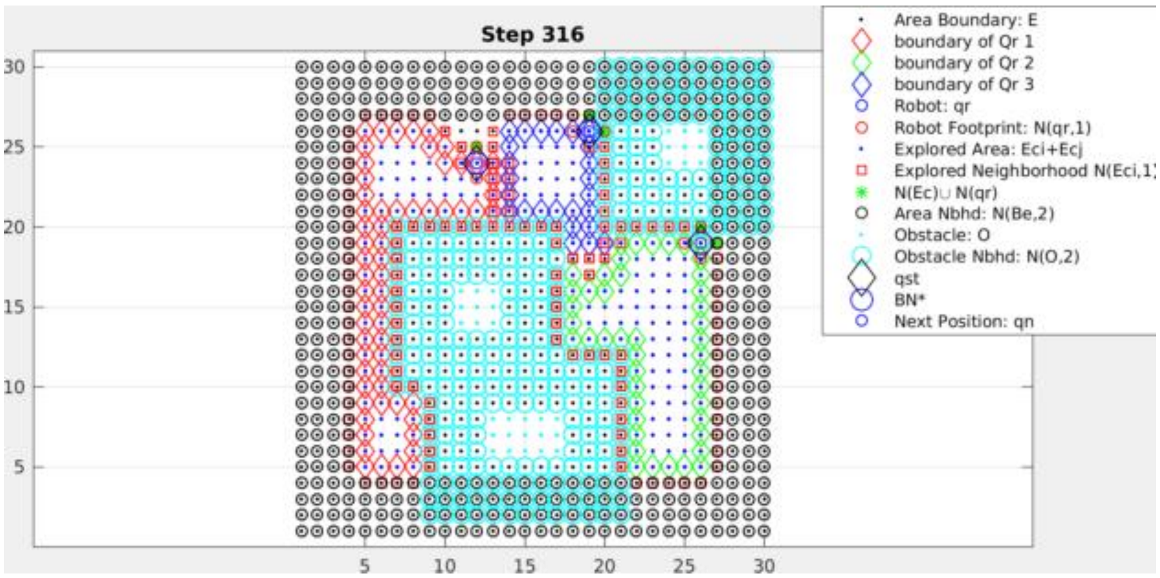


Figure 66. Coverage Algorithm Implemented by Three Robots in a One-Obstacle Environment

D. RESULTS AND ANALYSIS

In this part, we evaluate our results and analyze what are the reasons for affecting the time steps and the coverage.

1. Results of Experiments

Table 10 lists the number of steps each experiment took to implement the algorithm. The final time steps ratio is defined as

$$\frac{\textbf{Actual \# Steps}}{\textbf{Min \# Steps}},$$

where **Actual # Steps** is taken from Table 10 and **Min # Steps** is taken from Table 9. And, the result is shown in Table 11.

Table 10. Number of Steps Taken by Each Experiment

		Number of obstacles		
		1	2	3
number of robots	1	Experiment 1	Experiment 2	Experiment 3
		435	315	353
	2	Experiment 4	Experiment 5	Experiment 6
		334	328	259
	3	Experiment 7	Experiment 8	Experiment 9
		203	233	316

Table 11. Final Time Steps Ratio

		Number of obstacles		
		1	2	3
number of robots	1	1.38	1.35	2.80
	2	2.12	2.80	4.11
	3	1.93	2.99	7.52

2. Factors that Affect Time Steps

After evaluating the result from the experiments, the characteristics listed below are factors that I consider affecting how long it takes in the experiments.

a. Number of obstacles

Increasing the number of obstacles leads to a boost of neighborhood points, which generates a significant reduction in free spaces. Evidence is provided in Table 8. From Table 11, we find that as the number of obstacles increases, the time steps ratio gets higher. This means the efficiency diminished due to an increase in the number of obstacles.

b. Obstacles Connected to the Boundary

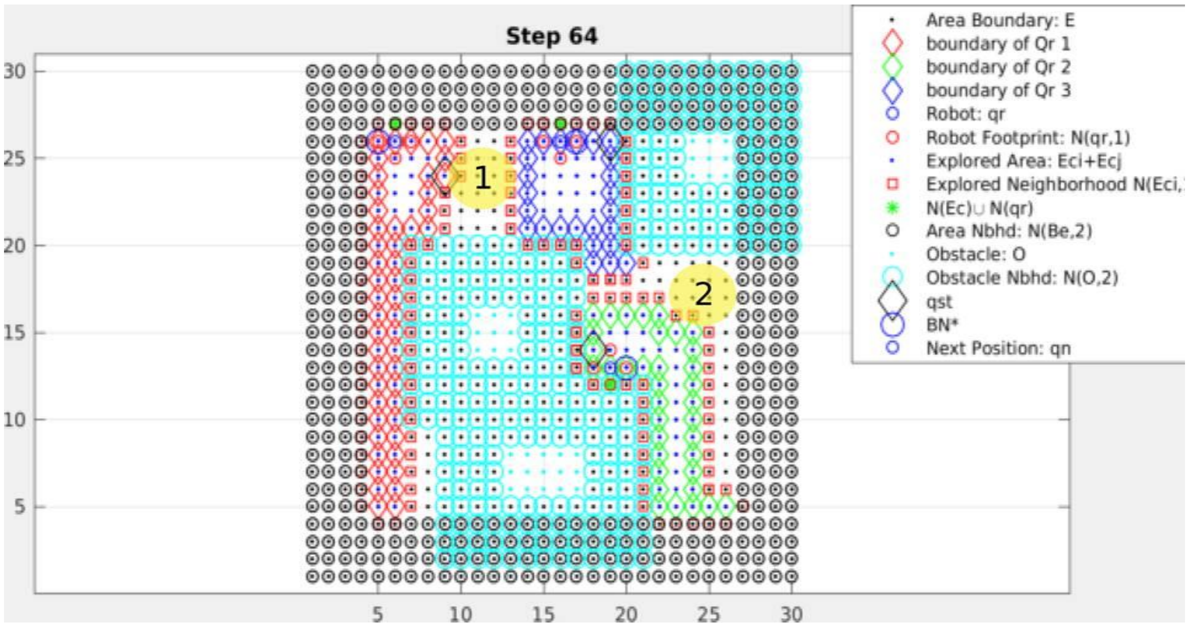
Having obstacles connected to the boundary causes a dramatic increase in the number of points on the boundary of the area explored by the robot. That is the reason why the number of free grid cells of the three-obstacle environment is much lower than in the environments with one and two obstacles. Consequently, this extends the steps a robot requires to patrol.

c. Number of Robots

Increasing the number of robots is a double-edged sword. The advantage of doing so lies in sharing the steps in the environment. A best-case scenario is when the domain area is broad enough that robots will not meet each other very often. The drawback occurs when robots meet each other very often, which increases the number of ignored free spaces due to the robots colleague avoidance behavior. Thus, it extends the steps directly. Furthermore, from the final time steps table, three robots have a higher ratio than the other two in an environment with the same number of obstacles. Take the environment with three obstacles as an example: the ratio of a single robot is 2.8, the ratio of two robots is 4.11, and the ratio of three robots is 7.52. This indicates that multiple robots actually have less efficiency than a single robot. It is worth noting that in two cases, experiment 5 and 9, the time steps they took are more than what we expected. In experiment 5, frequent updating $\mathbf{q}_{st}$ of robot 2 is the primary reason that leads to more time steps. And in experiment 9, frequent colleague avoidance behavior performed by three robots is the factor that extends the time steps.

d. Colleague Avoidance Behavior

Colleague avoidance is a very essential and indispensable behavior in the coverage algorithm of multi-robot systems. This feature limits the number of robots to perform the algorithm in different environments. In experiment 9, for example, three robots generated at least two ignored free spaces, shown in Figure 67, which further extended the steps and resulted in uncomplete coverage. Fewer free spaces with many robots lead to frequent colleague avoidance, and that is a problem resulting in poor performance.



Ignored unexplored spaces are marked as one and two.

Figure 67. Ignored Unexplored Spaces Due to Colleague Avoidance Behavior

e. *Uncertainty Factor*

From observation, a factor that extends the steps is unpredictable, which introduces uncertainty to the algorithm. If $\mathbf{q}_{st}$ is defined and visited earlier, the robot could cover the ignored unexplored area under clockwise travel. This would save many steps to complete the algorithm. Because the function of both counterclockwise and clockwise patrols is looking for an unexplored point, this makes one of the patrols redundant if complete coverage is not essential in the MCM mission.

3. Factor that Affects Complete Coverage

The number of ignored free spaces increases in a narrow environment with many robots. Since there are no more unexplored points for robots to visit in as far as the robot knows, due to colleague avoidance behavior, they execute the algorithm in wrap mode. Robots start to travel counterclockwise and search for unexplored points. If these points cannot be reached due to colleague avoidance before $\mathbf{q}_{st}$, clockwise patrol would be activated. If unexplored spaces still cannot be visited before $\mathbf{q}_{tr}$ for the same reason, the algorithm will be terminated. And, this leads to the coverage not being completed. Experiment 9 is the best example to illustrate this. Since it has three obstacles and three robots, fewer free points and more robots, this leads to frequent colleague avoidances among robots. And because experiment 9 has too many colleague avoidances, it is unable to complete the coverage. The results of complete coverage are presented in Table 12.

Table 12. Coverage Completeness Ratio

		Number of obstacles		
		1	2	3
number of robots	1	100.00%	100.00%	100.00%
	2	100.00%	100.00%	100.00%
	3	96.51%	100.00%	92.86%

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION

Path planning coverage algorithms for multi-robot systems could be applied to many fields. MCM is just one of them. Mine countermeasure missions are usually carried out by ships. With the development and advancement in the technology of control and multi-robot systems, though, it is believed that these robot agents will one day take over the MCM mission. As a result, the path planning coverage algorithm becomes an essential topic in this field because the combination of an efficient algorithm and accurate sensors saves time and human lives. Many path planning coverage algorithms for multi-robot systems have been developed, tested, and applied on agents on land. After studying the collected algorithms, we decided to implement one of them in an MCM environment. The aim of this thesis is to develop the algorithm in the MATLAB program and to evaluate whether the implemented algorithm is suitable in an MCM mission.

To construct a picture of cell decomposition, which is the foundation of path planning coverage, we introduced three classifications of cell decomposition in Chapter I. Approximate cellular decomposition does not depict targets in a region exactly since all targets are composed of grids. Robots in an approximate cellular environment usually have a footprint of the same size as the grid. The coverage is complete after all the cells are visited by the robots. The wavefront algorithm is quite a good example to convey the idea. Exact cell decomposition is defined as having the explicit shape of the targets. The cells are constructed based on the distribution of targets, and the robots move in back-and-forth motion within each cell. Trapezoidal decomposition gives a valuable illustration of exact cell decomposition. Another cell decomposition method, the semi-approximate method, assigns each cell equal width but allows them to vary in length vertically. Robots follow the depth-first rule to explore the area. Some boundaries of the cell will be covered twice while others may not be visited at all. And, that is how the semi-approximate method is defined.

Research on coverage algorithms of multi-robot systems is described in Chapter II. None of those studies had applied a coverage algorithm related experiment to a real AUV. As a result, considering the scenario, environment, characteristics of mission, and available knowledge applicable to our purpose from previous work, we chose [5] as our algorithm to implement in the MCM scenario.

In Chapter III, we first explained the algorithm from pseudo code. There are two modes under this algorithm. One is normal mode, in which robots operate when free space is reachable. The other is wrap mode, which is activated when no more free space can be reached. The free space consists of the points that remain after removing the boundary, the neighborhood of boundary, the obstacles, the neighborhood of obstacles and the neighborhood of robot's footprints, as well as the explored area of robots. In wrap mode, each robot travels along its own explored area in a counterclockwise direction. If no more free points can be reached, the robots go clockwise in order to look for free space. If there is no more free space, the algorithm is terminated. After analyzing the algorithm, we developed and performed the algorithm in a two dimension grid-based environment in the MATLAB program.

In Chapter IV, we simulated nine experiments with a different number of robots and obstacles in each experiment. The number of both robots and obstacles from one to three is distributed in each experiment. We found that increasing the number of obstacles results in a decreased amount of free space. But, having an obstacle connected to the domain boundary boosts the time step because the line along the obstacle will become part of the boundary explored by the robot. As a result, the robot has to move back and forth in wrap mode looking for free space. This feature is observed from experiment 3. Furthermore, the result shows that when the number of robots increases, efficiency is diminished primarily due to the behavior of colleague avoidance. This characteristic is obvious when there are many obstacles in the domain area. The outcome subverts my anticipation because of complexity of environment and robot's behavior of colleague avoidance.

A. PROS AND CONS

Considering the result of experiments, the comments below are my perspectives toward this algorithm.

1. Idea

Avoiding repeated paths in the coverage algorithm is not simple. It is a good idea to develop an algorithm that could complete the coverage with a minimum number of repeated paths. We predict the time step will be reduced by the development of a coverage algorithm with the fewest repeated paths. In this work, we implemented [5] and observed that this algorithm does not really save time steps. In fact, the time step rises since the robot has to move back and forth on the boundary of the explored area to look for free space. Sometimes the coverage could not be finished due to frequent colleague avoidance. Nevertheless, since the robot does not have prior knowledge about its surroundings, this result can be considered as an acceptable approximation.

2. Scalability

The domain area in this work is small in scale, only a 30 by 30 grid. Since the robots have no prior information about the environment, they have to memorize and store all the data after each step. As a result, much space would be required for storing information and the amount of time for each step would increase when this algorithm is operated under a large scale environment.

3. Completeness of Coverage

From the result of experiment 7, we observed that some free space within the boundary of area explored by the robot could not be completed due to colleague avoidance behavior. This might be improved by turning off the colleague avoidance behavior once the located free points are within the boundary of area explored by a robot.

B. FUTURE WORK

The followings are further researches which are supposed to be studied in the future.

1. Monte Carlo Simulation

Due to limited time, the initial positions of the robots were fixed in each trial. The objective outcome of this algorithm would be better weighed through quantitative analysis. Thus, random positions must be chosen for at least 100 trials for each scenario.

2. Improve the Algorithm

To complete the coverage as often as possible, the colleague avoidance behavior should be paused if a robot operating in wrap mode locates free points inside the boundary of its own explored area.

3. Run the Algorithm with Robotic Simulator

Running the algorithm with Gazebo or Stage will help us to narrow the gap between theory and reality. From the results of the simulator, we could improve the algorithm in the MATLAB program environment and inspect it in the simulator again. With this “back and forth” procedure, the behavior and experiment result of the robots would become much more like that of real robots.

4. Appliy the Algorithm on Land Robots in MCM-like Environment

Before testing the algorithm on an AUV, it is better to inspect the behavior and reaction of robots on land using an MCM-like environment. Through the examination, we could observe their performance, improve the algorithm if required, and anticipate their actions when they are operated under water. Most important, we could iterate the process in the least amount of time and at minimal cost compared with operating the robots directly in an underwater environment.

5. Implement the Algorithm on an AUV in a Littoral Zone

If the result and performance are good enough to indicate that this algorithm is applicable in an underwater environment, we could implement it on an AUV in a littoral zone. Through such an experiment, we could observe the amount of time that it takes to complete the algorithm. And, it would be perfect if the coverage could be finished. If not, the contributing factors would need to be determined and solved in order to improve the algorithm and performance of the robots.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] H. Choset, "Coverage for robotics–A survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113–126, 2001.

[2] N. Amato. (2004, Fall). Approximate Cell Decomposition Methods. [Online]. Available: https://parasol.tamu.edu/people/amato/Courses/padova04/lectures/L4.approx-decomp.pdf

[3] A. Zelinsky et al., "Planning paths of complete coverage of an unstructured environment by a mobile robot," in *Proceedings of International Conference on Advanced Robotics*, 1993, pp. 533–538.

[4] J. C. Latombe, *Robot Motion Planning*, 1st ed. Norwell, MA: Kluwer Academic, 1991, pp. 16–17.

[5] S. S. Ge and C. Fua, "Complete multi-robot coverage of unknown environments with minimum repeated coverage," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 715–720.

[6] M. J. Mataric, "Behaviour-based control: Examples from navigation, learning, and group behaviour," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 9, pp. 323–336, 1997.

[7] L. P. Reis and N. Lau. (n.d.). Intelligent Robotics. [Online]. Available: https://web.fe.up.pt/~lpreis/ir2007_08/documents/IR0708-3-AgentArchitectures.pdf. Accessed April 30, 2017.

[8] C. A. Berry. (2006–2007, Spring). ECE497: Introduction to Mobile Robotics Lecture 6. [Online]. Available: http://www.rose-hulman.edu/~berry123/Courses/ECE425/Spring07_files/Lecture6%20-%20Types%20of%20Control%20Architectures.pdf

[9] Z. J. Butler, A. A. Rizzi, and R. L. Hollis. (2000). Complete distributed coverage of rectilinear environments. [Online]. Available: https://www.cs.cmu.edu/~motionplanning/papers/sbp_papers/integrated1/butler_coverage.pdf

[10] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and Service Robotics*, 1$_{st}$ ed. London, UK: Springer, 1998, pp. 203–209.

[11] C. Stachniss and W. Burgard, “Exploring unknown environments with mobile robots using coverage maps,” in *18_{th} International Joint Conference Artificial Intelligence. IJCAI*, Acapulco, Mexico, 2003, pp. 1127–1134.

[12] M. A. Batalin and G. S. Sukhatme, “Spreading out: A local approach to multi-robot coverage,” in *Distributed Autonomous Robotic Systems 5*. Japan: Springer, 2002, pp. 373–382.

[13] C. S. Kong, N. A. Peng, and I. Rekleitis, “Distributed coverage with multi-robot system,” in *Proceedings 2006 IEEE International Conference on Robotics and Automation*. ICRA, 2006, pp. 2423–2429.

[14] N. Hazon, F. Mieli, and G. A. Kaminka, “Towards robust on-line multi-robot coverage,” in *Proceedings 2006 IEEE International Conference on Robotics and Automation*. ICRA, 2006, pp. 1710–1715.

[15] M. J. Bays et al., “Theory and experimental results for the multiple aspect coverage problem,” *Ocean Engineering*., vol. 54, pp. 51–60, 2012.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California